HZ Books
华章教育

PEARSON

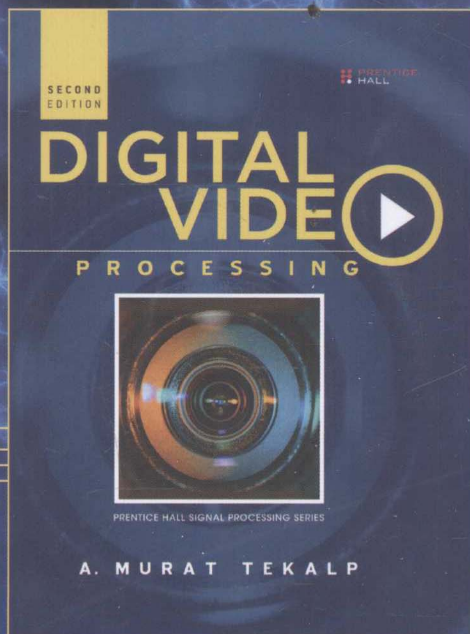
国外电子与电气工程技术丛书

# 数字视频处理

## （英文版·第2版）

[土耳其] A. 缪拉·泰卡尔普（A. Murat Tekalp） 著

*Digital Video Processing*
*Second Edition*




机械工业出版社
China Machine Press

多年来，《Digital Video Processing》都是无数工科学生和专业人士深入学习数字图像和视频处理技术的权威指南。在《Digital Video Processing》第2版中，作者对图像处理、计算机视觉、视频压缩等领域的重大发展进行了探讨，也对诸如数字电影、超高分辨率视频、3D视频等新应用进行介绍。

全书内容详尽、组织均衡、论述严谨，全面覆盖了图像滤波、运动估计、跟踪、分割、视频滤波和压缩等诸多方向。书中对各章节的习题都进行了更新，并加入了新的MATLAB项目，已使本书成为一本全新的教材。

## 内容包括：

- 多维信号与系统：转换、采样、格式转换。
- 数字图像和视频：人类视觉、数字视频、视频质量。
- 图像滤波：梯度估计，边缘检测，尺度缩放，多分辨率表示、增强、去噪、复原。
- 运动估计：成像，运动模型，有差分法、匹配法、优化法、变换域方法，3D运动与形状估计。
- 视频分割与跟踪：色彩与运动分割、变化检测、镜头边界检测、视频抠图、视频跟踪与性能评估。
- 视频滤波：运动补偿滤波，多帧标准转换，多帧噪声过滤、复原，超分辨率重建。
- 图像压缩： JPEG、小波、JPEG 2000。
- 视频压缩：早期标准、ITU-T H.264/ MPEG-4 AVC、HEVC、可扩展视频压缩、立体视觉和多视图法。

## 作者简介：

**A. 缪拉·泰卡尔普（A. Murat Tekalp）** 是Koc大学教授、IEEE会士、欧洲学术研究院和土耳其科学院的院士。他是《Proceeding of the IEEE》的编委会成员和ICT的欧洲委员会国际专家，曾在美国罗切斯特大学（纽约）任教18年，并成为杰出教授。2004年，他获得了土耳其最高科学奖：TUBIAK科学奖。泰卡尔普博士曾担任IEEE图像与多维信号处理技术委员会的主席，也是IEEE多媒体信号处理技术委员会的创始成员，还曾担任欧洲信号处理协会（EURASIP）的《Signal Processing: Image Communication》杂志（由Elsevier出版）的主编。有美国伦斯勒理工学院的电气、计算机和系统工程学科领域博士学位。








上架指导：数字视频

ISBN 978-7-111-53286-6




PEARSON
www.pearson.com

投稿热线：(010) 88379604
客服热线：(010) 88378991 88361066
购书热线：(010) 68326294 88379649 68995259

华章网站：www.hzbook.com
网上购书：www.china-pub.com
数字阅读：www.hzmedia.com.cn

封面设计·锡彬

定价：99.00元

国外电子与电气工程技术丛书

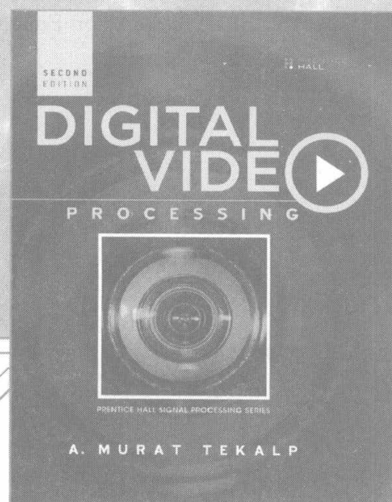
# 数字视频处理

## （英文版·第2版）

[土耳其] A. 缪拉·泰卡尔普（A. Murat Tekalp） 著

*Digital Video Processing*

*Second Edition*





机械工业出版社
China Machine Press

**图书在版编目（CIP）数据**

数字视频处理（英文版·第2版）/（土）泰卡尔普（Tekalp, A. M.）著．—北京：机械工业出版社，2016.4
（国外电子与电气工程技术丛书）
书名原文：Digital Video Processing (Second Edition)

ISBN 978-7-111-53286-6

I. 数… II. 泰… III. 数字视频系统–数字信号处理–英文 IV. TN941.3

中国版本图书馆CIP数据核字（2016）第055446号

**本书版权登记号：图字：01-2016-0767**



出版发行：机械工业出版社（北京市西城区百万庄大街22号 邮政编码：100037）

责任编辑：张梦玲 责任校对：董纪丽

印　　刷：北京市荣盛彩色印刷有限公司 版　　次：2016年4月第1版第1次印刷

开　　本：186mm×240mm 1/16 印　　张：34.5

书　　号：ISBN 978-7-111-53286-6 定　　价：99.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

客服热线：(010) 88378991 88361066 投稿热线：(010) 88379604

购书热线：(010) 68326294 88379649 68995259 读者信箱：hzjsj@hzbook.com

# 出版者的话

文艺复兴以来，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的传统，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，信息学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘划了研究的范畴，还揭示了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的信息产业发展迅猛，对专业人才的需求日益迫切。这对我国教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短的现状下，美国等发达国家在其信息科学发展的几十年间积淀和发展的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀教材将对我国教育事业的发展起到积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章公司较早意识到“出版要为教育服务”。自1998 年开始，我们就将工作重点放在了遴选、移译国外优秀教材上。经过多年的不懈努力，我们与Pearson、McGraw-Hill、Elsevier、 John Wiley & Sons、CRC、Springer等世界著名出版公司建立了良好的合作关系，从他们现有的数百种教材中甄选出Thomas L. Floyd、Charles K. Alexander、Behzad Razavi、John G. Proakis、Stephen Brown、Allan R. Hambley、Albert Malvino、Mark I.Montrose、David A. Johns 、Peter Wilson 、H. Vincent Poor、Dikshitulu K. Kalluri、Bhag Singh Guru、Stephane Mallat等大师名家的经典教材，以“国外电子与电气工程技术丛书”为总称出版，供读者学习、研究及珍藏。这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍。其影印版“经典原版书库”作为姊妹篇也越来越多被实施双语教学的学校所采用。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证。随着电气与电子信息学科建设的不断完善和教材改革的逐渐深化，教育界对国外电子与电气工程教材的需求和应用都将步入一个新的阶段，我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。华章公司欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：


华章网站：www.hzbook.com

电子邮件：hzit@hzbook.com

联系电话：(010) 88379604

联系地址：北京市西城区百万庄南街1号

邮政编码：100037





华章科技图书出版中心

# 前　言

本书于 1995 年出版了第 1 版，是一本全面介绍数字视频处理的教材。其中根据视频处理领域的重要论题分成了 25 章，在一个学期的课程中，每章可以用一到两次课进行讲授。那个时期的数字视频技术和视频处理算法还不太成熟，数码摄像机和 DVD 刚刚商业化，数字电视标准正在制定，而数字电影则还没有纳入议程。因此，与当今的技术水平相比，第 1 版提到的一些方法 / 算法和技术已经过时，比如像素级递归的运动估计、矢量量化、不规则形状压缩、基于模型的编码等已不再先进，还有一些诸如模拟视频 / 电视和 128K 可视电话等技术则已经淘汰了；同时近 20 年来此领域的重大进展显然也无法体现出来。

第 1 版出版至今已有 20 多年了，在当今这个数字化时代，数字视频已广泛应用于我们的日常生活。信号处理与计算机视觉领域的重大发展促进了视频处理算法的不断成熟，能够应用于不同用途的最常用又有效的算法与技术也更加清晰。因此，现在是本书推出新版的最好时机。本书围绕图像与视频处理的最新发展进行了精心编排，力图成为一本内容全面、结构严谨的教材。

第 2 版大幅度改进了内容与表述的组织方式，包含当今最先进的技术、最有效的算法和最新的知识。全书共分 8 章，每章对应一个主题，分别是多维信号与系统、数字图像和视频、图像滤波、运动估计、视频分割与跟踪、视频滤波、图像压缩、视频压缩等，每个主题侧重介绍最有效的技术。与第 1 版相比，本版不是简单的内容增补，而是一次全新的改写。

本书可作为高年级本科生或研究生的数字图像与视频处理课程的教材，要求读者预先掌握微积分、线性代数、概率论和一些基本的数字信号处理概念。具有计算机科学背景但不熟悉信号处理基本概念的读者可以跳过第 1 章，从第 2 章开始学习。尽管本书表述严谨，但仍然像一般教材一样从原理开始讲起，因此也可以用作产业界或学术界的工程师和研究人员自学的参考书。本书可帮助读者理解图像和视频处理方法的理论基础；学习用最常用、最有效的算法解决常见的图像与视频处理问题；通过每章最后的习题集和 MATLAB 项目，可加深对知识的理解和方法的掌握。

数字视频处理就是对数字视频比特流的各种操作。所有的数字视频应用都离不开压缩。此外，为了获得高质量图像或提取特定信息，数字视频应用也离不开广泛应用于格式转换、增强、复原、超分辨率重建等场合的滤波处理；有些应用还需要用到其他的处理，以实现运动估计、视频分割和 3D 场景分析。视频的帧与帧之间存在着大量的时

间相关性（冗余），这使得视频处理不同于静态图像处理。可以将视频看成是静态图像序列，并逐帧独立处理；但若采用基于帧间相关性的多帧联合处理技术，我们能够开发出更有效的视频处理算法，例如运动补偿滤波和预测。此外，某些任务，比如运动估计或动态场景分析，显然是无法基于单个图像来进行的。

本书的目的是为读者提供图像（单帧）和视频（多帧）处理方法的数学基础。特别是，本书还回答了以下基本问题：

- 如何从噪声中分离出图像（信号）？
- 内插、复原和超分辨率重建之间是否有内在的联系？
- 对于不同的应用，该如何估计 2D 和 3D 运动？
- 如何将图像和视频分割成感兴趣的区域？
- 如何跟踪视频中的对象？
- 与图像滤波相比，视频滤波问题是否更趋向于适定？
- 超分辨率重建为何能够实现？
- 能否从视频片段中得到高质量的静态图像？
- 图像和视频压缩为什么能够实现？
- 如何压缩图像和视频？
- 图像 / 视频压缩的最新国际标准是什么？
- 3D 视频表现和压缩的最新标准是什么？

图像和视频处理问题大都是病态的（欠定的和 / 或对噪声敏感的），并且它们的解都依赖于某些图像和视频模型。在附录 A 中讨论了用于病态问题解的图像建模方法。实际上，图像模型可以分成基于局部平滑的、基于变换域稀疏的和基于非局部自相似的等种类。

图像处理算法大都使用了以上模型中的一种或多种。此外，视频模型还包括基于全局平移或块运动、参数化运动、运动（空间上）的平滑性、时域运动单调性（时域连续或平滑）、3D 空 – 时频域的平面支撑等种类。

各章概述如下：

第 1 章回顾了多维信号、变换和系统的基础知识，它们是许多图像和视频处理方法的理论基础。我们还介绍了空 – 时采样的体制（如逐行和隔行采样），以及采样格式转换理论。读者如果具有计算机科学背景而只是不熟悉信号处理概念，则可以跳过本章，直接从第 2 章开始学习。

第 2 章给出了数字图像与视频的基础知识，主要内容包括人类视觉、空间频率、彩色模型、模拟和多视角视频表示、数字视频质量评估等基本概念，以及一些常见的数字视频应用，如数字电视、数字电影和互联网视频流等。

第3章介绍图像（静止帧）滤波类问题，比如图像重采样（抽取与内插）、梯度估计与边缘检测、增强、去噪、复原等。还介绍了线性移不变滤波器、自适应滤波器和非线性滤波器。附录A中给出了求解病态逆问题的一般性框架。

第4章介绍2D和3D的运动估计方法。运动估计是数字视频处理的核心，因为运动是视频的显著特征，并且运动补偿滤波是利用时间冗余的最有效的方法。再者，许多计算机视觉工作的第一步都是2D或3D的运动估计与跟踪。2D运动估计一般分为稠密光流估计或稀疏特征对应估计两类，可以基于参数法和非参数法来实现。非参数法包括基于图像梯度的光流估计法、块匹配法、像素递归法、贝叶斯法和相位相关法。基于仿射模型或单应性的参数法可以用于图像配准或局部变形估计。3D运动/结构估计法一般都基于双帧极线约束法（主要是针对立体对的）或多帧因子分解法。欧氏3D结构重建需要对所有相机进行标定，而投影重建法则可以无需标定。

第5章介绍图像分割和变化检测，以及基于参数聚类法和贝叶斯法的主要运动或复杂运动分割。我们还讨论了运动估计与分割的同时实现问题。因为双视角运动估计技术对于图像梯度或对应点的估计精度很敏感，而对于单视角长序列对，其分割对象的运动跟踪结果更鲁棒，所以我们也对它们进行了相关讨论。

第6章介绍视频滤波，包括标准转换、去噪和超分辨率重建等内容。首先介绍了运动补偿滤波的基本原理，随后介绍了标准转换问题，包括帧速转换和去隔行等。视频帧的画面中经常存在颗粒，尤其在静止帧模式下观看时更加严重。为此，讨论了用于噪声抑制的运动自适应和运动补偿滤波。最后介绍了一种统一各种视频滤波问题的综合模型，可用于低分辨率视频获取和超分辨率重建。

第7章介绍包括二值图像（传真）和灰度图像在内的静态图像压缩方法与标准，如JPEG和JPEG 2000等。还特别讨论了有损的离散余弦变换编码和小波变换编码等方法。

第8章讨论视频压缩方法和标准，它们是实现数字电视、数字电影等数字视频应用的基础。在简要介绍视频压缩的不同方法后，详细描述MPEG-2、AVC/H.264和HEVC等标准，以及这些标准在可伸缩视频编码和立体/多视角视频编码方面的扩展。

本教材是近20多年来我在数字图像与视频处理领域的教学结晶。本书内容丰富、组织严谨，全面覆盖了图像滤波、运动估计与跟踪、图像/视频分割、视频滤波、图像/视频压缩等方面的基本原理和最新成就。然而，一本教材无法覆盖数字视频处理和计算机视觉领域所有的最新成就，因此本书只对最基本、最常用的技术和算法加以详解，而对更多的先进算法和最新研究成果只进行简介，并提供用于自学的参考文献。每章最后都包含问题集和MATLAB项目，以便读者对所学到的方法进行练习。

教师可以通过申请获得教学资料。根据各校的课时安排可在一个学期的数字图像与视频处理课程中讲完本书的全部内容。另一种方式是将本书内容分到两个学期中，这样

就有更多的时间对每个主题的细节进行探讨：第一学期可以开设数字图像处理课程，讲解第 1 ~ 3 章、第 7 章的内容；第二学期后续开设数字视频处理课程，讲解第 4 ~ 6 章、第 8 章的内容。

显然，本书是信号处理和计算机科学相关组织研究成果的荟萃。每章都有很多引用并列出了相关参考文献，但肯定无法涵盖图像与视频领域科研与工业部门杰出研究者的所有成就。此外，对 ISO 和 ITU 组织中各位科学家经多年工作取得的图像与视频编码的显著成果，在这里也难以一一致意。

最后，衷心感谢 Xin Li（美国西弗吉尼亚大学，WVU）、Eli Saber、Moncef Gabbouj、Janusz Konrad 和 H.Joel Trussell 在本书成稿过程中的贡献。同时感谢 Prentice Hall 出版社的 Bernard Goodwin、Kim Boedigheimer 和 Julie Nahil 的帮助与支持。

——A. Murat Tekalp

于土耳其，伊斯坦布尔，Koc 大学

# Contents

# 目　录

## 第 4 章 运动估计 191

## 附录 C 优化方法 511



## 附录 D 模型拟合 521



## 术语表 527

CHAPTER 1

# Multi-Dimensional Signals and Systems

*There are some fundamental differences between the theory of one-dimensional and multi-dimensional signal processing that arise from the related facts that zeros and poles of the multi-dimensional z-transform are not isolated points but functions, and multi-variable polynomials in general do not factor.*

The theory of multi-dimensional (MD) signals and systems is fundamental to understanding digital image and video processing. Digital images are two-dimensional (2D) sequences (partially ordered) with two discrete spatial variables; they can also be represented by arrays (vectors or matrices). Digital video is a three-dimensional (3D) function of two spatial and one temporal variable, and 3D video is a four-dimensional (4D) function of three spatial and one temporal variable. Digital filters to process these signals are MD systems. MD transforms help to understand spatial and temporal frequency concepts, and physical and normalized frequency variables. This chapter introduces the fundamental concepts of MD signals, transforms, and systems, as well as sampling MD analog signals on lattices and sampling structure conversion, which are essential for digital image and video processing.

There are some fundamental differences between the theory of 1D and MD signal processing, which arise from: i) zeros and poles of the MD z-transform are not

isolated points but functions, and ii) multi-variable polynomials in general do not factor. As a result, some 1D signal-processing methods do not readily generalize to multi dimensions, but at the same time, it is possible to develop new algorithms in multi dimensions that do not have 1D counterparts. In the following, we present the main results for MD signals, systems, and transforms in 2D notation, since it is easier to visualize in 2D. Most definitions and results for 2D signal processing can be readily generalized to more than two dimensions, although the notation becomes more cumbersome and abstract and visualization may not be possible.

## 1.1 Multi-Dimensional Signals

An MD signal is a multi-variable function or sequence of $M \geq 2$ independent continuous, discrete, or mixed variables.

**Definition:** An analog MD signal $s(\mathbf{x}) = s(x_1, x_2, \ldots, x_M)$ is a function of $M$ continuous variables, where $\mathbf{x} = [x_1\, x_2 \ldots x_M]^{\mathrm{T}}$. The function $s(\mathbf{x})$ may be scalar valued (e.g., gray-scale image) or vector valued (e.g., color image).

**Definition:** A discrete MD signal $s(\mathbf{n}) = s(n_1, n_2, \ldots, n_M)$ is an MD sequence defined over a lattice, which is a partially ordered set of $M$-tuples of integers $\mathbf{n} = [n_1\, n_2 \ldots n_M]^{\mathrm{T}}$. Again, $s(\mathbf{n})$ may be scalar or vector valued.

**Definition:** A mixed MD signal is a function of $M$ variables some of which are continuous, while others are discrete. An analog-video signal is a mixed signal.

### 1.1.1 Finite-Extent Signals and Periodic Signals

This section discusses MD finite-extent signals and MD periodic signals that are isomorphic to each other.

#### Finite-Extent Signals

Since cameras have finite-size sensors and video is recorded over a finite duration, most MD signals of interest are finite extent; i.e., they are defined only within a finite region in space/time, called the "support." The support of a signal may have an MD rectangular or arbitrary shape.

A 2D signal is said to have wedge support if it is defined only within two lines emanating from the origin, as shown in Figure 1.1(d). Quarter-plane (Figure 1.1(a)), half-plane (Figure 1.1(b)), and non-symmetric half-plane (NSHP) (Figure 1.1(c)) supports are special cases of wedge support.

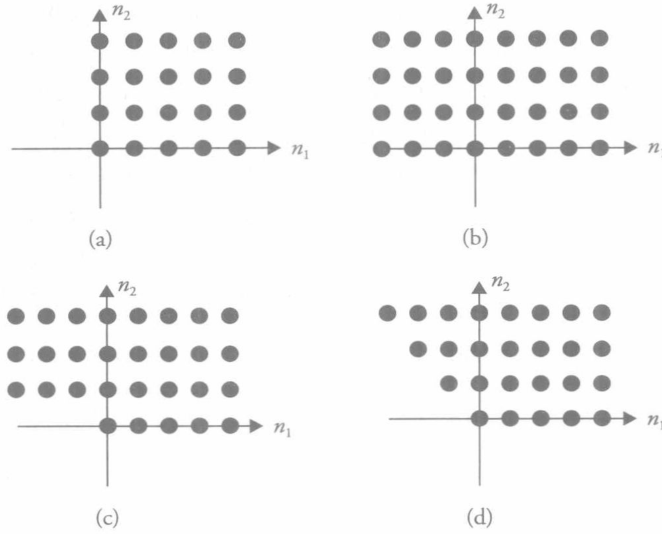



**Figure 1.1** Support of 2D signals: (a) quarter-plane support; (b) half-plane support; (c) non-symmetric half-plane support; and (d) wedge support.

**Example.** Digital images have finite quarter-plane support, which is depicted in Figure 1.1(a), i.e., they are defined within a rectangle $0 \leq n_1 \leq N_1 - 1$ and $0 \leq n_2 \leq N_2 - 1$, where $N_1$ and $N_2$ denote the horizontal and vertical size of an image in pixels.

### Periodic Signals

An MD sequence $\tilde{s}(\mathbf{n})$, where $\mathbf{n} = [n_1\ n_2 \ldots\ n_M]^{\mathrm{T}}$, is said to be periodic with the periodicity matrix $\mathbf{N}$ if it satisfies

$$\tilde{s}(\mathbf{n}) = \tilde{s}(\mathbf{n} + \mathbf{N}\mathbf{r}) \tag{1.1a}$$

for all integer-valued $M$-vectors $\mathbf{r}$, where $\mathbf{N}$ is an $M \times M$ periodicity matrix, such that $\det(\mathbf{N}) \neq 0$.

For 2D signals $\mathbf{r} = \begin{bmatrix} r_1 \\ r_2 \end{bmatrix}$ and $\mathbf{N} = [\mathbf{n}_1 \mid \mathbf{n}_2] = \begin{bmatrix} N_{11} & N_{12} \\ N_{21} & N_{22} \end{bmatrix}$, and the vector equation (1.1a) can be expressed in scalar form as

$$\begin{aligned} s(n_1, n_2) &= s(n_1 + r_1 N_{11}, n_2 + r_1 N_{22}) \text{ with } r_2 = 0 \\ &= s(n_1 + r_2 N_{12}, n_2 + r_2 N_{22}) \text{ with } r_1 = 0 \end{aligned} \tag{1.1b}$$

The integer-valued vectors $\mathbf{n}_1 = (N_{11}, N_{21})$ and $\mathbf{n}_2 = (N_{12}, N_{22})$ denote displacements from any sample in one period to the corresponding samples in two other periods. In other words, the 2D signal repeats itself at all integer multiples of the shift vectors $\mathbf{n}_1$ and $\mathbf{n}_2$ as depicted in Figure 1.2. The period defines a unit cell in the $(\mathbf{n}_1, \mathbf{n}_2)$ plane, which repeats over the whole plane. We note that $\mathbf{N}$ is not unique; two arbitrarily chosen linearly independent vectors $\mathbf{n}_1$ and $\mathbf{n}_2$ that point to the same sample in two different periods can be used to represent the same periodicity pattern.

A special case arises when the periodicity matrix is diagonal, given by $\mathbf{N} = \begin{bmatrix} N_1 & 0 \\ 0 & N_2 \end{bmatrix}$, which is illustrated in Figure 1.2(b). In this case, the shift vectors $\mathbf{n}_1$ and $\mathbf{n}_2$ lie along the horizontal and vertical axes. Then, $s(n_1, n_2)$ is said to be rectangularly periodic and satisfies

$$s(n_1, n_2) = s(n_1 + N_1, n_2) = s(n_1, n_2 + N_2) \text{ for all } (n_1, n_2)$$

**Example.** A 2D discrete complex exponential signal

$$s(n_1, n_2) = e^{j(\omega_1 n_1 + \omega_2 n_2)}$$

is rectangularly periodic in $(n_1, n_2)$ with period $(N_1, N_2)$ if

$$\frac{\omega_1}{2\pi} = \frac{k_1}{N_1} \text{ and } \frac{\omega_2}{2\pi} = \frac{k_2}{N_2}$$

where $N_1$, $N_2$, $k_1$ and $k_2$ are unitless integers, and the units of $\omega_1$ and $\omega_2$ are radians.

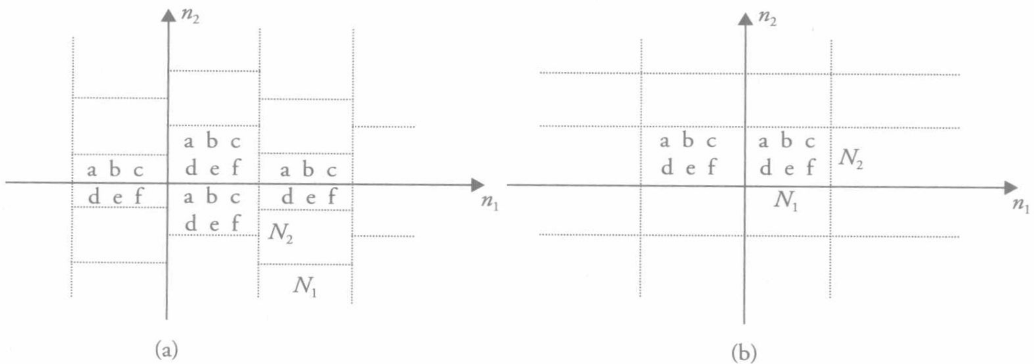



**Figure 1.2** Periodicity in 2D: (a) general periodicity defined by vectors $\boldsymbol{n}_1$ and $\boldsymbol{n}_2$ and (b) rectangular periodicity.

Periodic signals and finite-extent signals are isomorphic to one another, since given a rectangularly periodic signal, the main period of the periodic signal can be defined as a finite-extent signal, and given a finite-extent signal we can define a periodic signal through periodic extension

$$\tilde{s}(n_1, n_2) = \sum_{i_1=-\infty}^{\infty} \sum_{i_2=-\infty}^{\infty} s(n_1 - i_1 N_1, n_2 - i_2 N_2) \tag{1.2}$$

## 1.1.2 Symmetric Signals

There are multiple forms of symmetry in two or more dimensions. Most common forms of 2D symmetry are two-fold, four-fold, or circular symmetry. Symmetric signals can be finite or infinite extent. In the following, we define symmetry with respect to origin. Symmetry with respect to an arbitrary point can be defined by an appropriate shift of the coordinates.

### Two-Fold Symmetry

Two-fold rectangular symmetry is also called non-symmetric half-plane symmetry and is given by

$$s(n_1, n_2) = s(-n_1, -n_2) \tag{1.3a}$$

The distinct coefficients are depicted by dots in Figure 1.1(c). The remaining coefficients to complete the square support are determined by the symmetry.

### Four-Fold Symmetry

A more strict form of symmetry is the four-fold rectangular symmetry given by

$$s(n_1, n_2) = s(-n_1, n_2) = s(n_1, -n_2) \tag{1.3b}$$

The support of the distinct coefficients in this case is a quarter-plane.

### Circular Symmetry

A signal $s(n_1, n_2)$ is circularly symmetric if it is only a function of distance $n_1^2 + n_2^2$ from the origin. Circular symmetry implies four-fold symmetry.

## 1.1.3 Special Multi-Dimensional Signals

Some special signals play an important role in understanding image- and video-processing filters. They are separable signals, spatial-frequency patterns, MD impulse, and MD unit step signals, which are introduced in the following.

### Separable Signals

An MD signal (function) is separable if

$$s(n_1, n_2, \dots, n_M) = s_1(n_1)\, s_2(n_2) \dots s_M(n_M) \tag{1.4}$$

We can make the following observation in the case of 2D signals. A finite-support 2D signal $s(n_1, n_2)$ can be represented by a matrix $\mathbf{S}$. If the signal is separable, then the matrix $\mathbf{S}$ can be written as the outer product $\mathbf{S} = \mathbf{s}_1 \mathbf{s}_2^T$, where the vectors $\mathbf{s}_1$ and $\mathbf{s}_2$ denote samples of 1D signals $s_1(n_1)$ and $s_2(n_2)$, respectively. We note that while a general $N_1 \times N_2$ matrix has $N_1 N_2$ degrees of freedom, the outer product has $N_1 + N_2$ degrees of freedom.

**Example.** A 2D discrete complex exponential signal is separable, since

$$s(n_1, n_2) = e^{j(\omega_1 n_1 + \omega_2 n_2)} = e^{j\omega_1 n_1} e^{j\omega_2 n_2} = s_1(n_1) s_2(n_2)$$

### Spatial-Frequency Patterns

We can define the horizontal spatial-frequency pattern

$$s(n_1, n_2) = \cos(\omega_1 n_1)$$

which is the same for all rows (image lines), or the vertical spatial-frequency pattern

$$s(n_1, n_2) = \cos(\omega_2 n_2)$$

which is the same for all columns, or the spatial-frequency pattern with 45-degree angular orientation

$$s(n_1, n_2) = \cos(\omega\, (n_1 - n_2))$$

### MD Impulse

An MD impulse is defined by

$$\delta(n_1, n_2, \cdots, n_M) = \begin{cases} 1 & n_1 = n_2 = \cdots = n_M = 0 \\ 0 & \text{otherwise} \end{cases} \tag{1.5a}$$

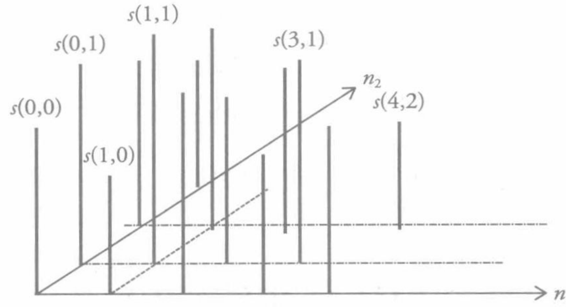



**Figure 1.3** Two-dimensional signal representation in terms of 2D impulses.

A 2D impulse denotes a point source with unity amplitude on the plane. Any discrete image (2D signal) can be expressed as a sum of shifted and weighted impulses as

$$s(n_1, n_2) = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} s[k_1, k_2]\delta(n_1 - k_1, n_2 - k_2) \tag{1.5b}$$

which is a generic 2D signal representation used in some derivations. This is illustrated in Figure 1.3.

**Example.** The 2D impulse is separable, since we can write

$$\delta(n_1, n_2) = \delta_1(n_1)\delta_1(n_2)$$

where $\delta_1(n_1) = \begin{cases} 1 & n_1 = 0 \\ 0 & \text{otherwise} \end{cases}$ is the 1D Kronecker delta function.

### MD Unit Step

An MD unit step is often used to indicate the support of an MD signal or the impulse response of an MD system. It can be defined on any support. For example, referring to Figure 1.1, we can define a 2D unit step with quarter-plane, half-plane, or wedge support.

**Example.** 2D unit step with quarter-plane support is defined as

$$u_{QP}(n_1, n_2) = \begin{cases} 1 & n_1 \geq 0,\ n_2 \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

2D unit step with half-plane support is defined as

$$u_{HP}(n_1, n_2) = \begin{cases} 1 & n_2 \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

The sample values are "1" on the dots shown in Figure 1.1(a) and (b), respectively, and zero elsewhere.

## 1.2 Multi-Dimensional Transforms

Signals, just like vectors, can be expanded onto some basis functions. The MD continuous (discrete) Fourier transform expands MD continuous (discrete) signals onto MD continuous (discrete) complex exponentials, which form an orthogonal basis. Here, we only present 2D Fourier transform for notational simplicity; however, all definitions and properties of 2D Fourier transform generalize to MD Fourier transform. Our discussion of 2D wavelet transform is delayed until we introduce finite impulse response (FIR) image filtering and multi-resolution/multi-scale image representations in Chapter 3.

### 1.2.1 Fourier Transform of Continuous Signals

While digital images and video are functions of discrete variables (sequences), we start this section with the Fourier transform of functions of continuous variables to introduce the notation used in Section 1.4, since signals with discrete variables are obtained by sampling signals with continuous variables.

**Definition:** The Fourier transform of a 2D continuous signal $s(x_1, x_2)$ is given by

$$S(u_1, u_2) = \iint_{(x_1, x_2)} s(x_1, x_2) e^{-j(u_1 x_1 + u_2 x_2)} \, dx_1 dx_2 \tag{1.6a}$$

and the inverse 2D Fourier transform is given by

$$s(x_1, x_2) = \frac{1}{(2\pi)^2} \iint_{(u_1, u_2)} S(u_1, u_2) \, e^{j(u_1 x_1 + u_2 x_2)} \, du_1 du_2 \tag{1.6b}$$

where $u_1$ and $u_2$ denote real spatial-frequency variables with units cycles/distance. We can also have a temporal variable, e.g., $s(x_1, x_2, t)$. Then, the unit for temporal frequency is Hz.

The 2D Fourier transform is complex:

$$S(u_1, u_2) = |S(u_1, u_2)| \, e^{j\theta(u_1, u_2)} = S_R(u_1, u_2) + jS_I(u_1, u_2) \tag{1.6c}$$

where $S_R(u_1, u_2)$ and $S_I(u_1, u_2)$ denote the real and imaginary parts, respectively, and $|S(u_1, u_2)|$ and $\theta(u_1, u_2)$ denote the Fourier magnitude and Fourier phase, respectively.

### Convergence of the Fourier Transform

The 2D signal $s(x_1, x_2)$ may have an infinite extent. Hence, we need to consider the conditions under which the double integral (1.6a) exists:

- *Uniform convergence:* The Fourier transform exists and is a continuous function of $u_1$ and $u_2$ (i.e., the double integral converges uniformly) if $\iint_{x_1 x_2} |s(x_1, x_2)|\, dx_1\, dx_2 < \infty$; i.e., $s(x_1, x_2)$ is absolutely integrable.
- *Mean-square convergence:* If $S(u_1, u_2)$ exists but has discontinuities, then a weaker form of convergence, called mean-square convergence, applies. For example, $s(x_1, x_2) = (\sin x_1)/x_1 \ (\sin x_2)/x_2$ is not absolutely integrable, but its Fourier transform converges in the mean-square sense. Then, we observe the Gibbs effect around points of discontinuity.
- *Generalized convergence:* In this case, neither uniform nor mean-square convergence applies, but $S(u_1, u_2)$ may still be defined using the Dirac delta function $\delta(u_1, u_2)$. For example, $s(x_1, x_2) = 1$ for all $(x_1, x_2)$ is not absolutely integrable, but its Fourier transform is defined in the generalized sense as $S(u_1, u_2) = \delta(u_1, u_2)$.

### Properties of Multi-Dimensional Fourier Transform of Continuous Signals

This section discusses some unique properties of the MD Fourier transform that do not have 1D counterparts.

#### *Decomposition into 1D Transforms*

MD Fourier transform can be decomposed into a series of 1D Fourier transforms over each variable separately, which is demonstrated here for the case of 2D. Eqn. (1.6a) can be rewritten as

$$S(u_1, u_2) = \iint_{(x_1, x_2)} s(x_1, x_2) e^{-ju_1 x_1} e^{-ju_2 x_2}\, dx_1 dx_2$$

$$= \int_{x_2} \left\{ \int_{x_1} s(x_1, x_2) e^{-ju_1 x_1}\, dx_1 \right\} e^{-ju_2 x_2}\, dx_2$$

$$= \int_{x_2} S(u_1, x_2) e^{-ju_2 x_2}\, dx_2$$

which shows that it can be evaluated as a 1D transform over $x_1$ followed by another 1D transform of the intermediate result $S(u_1, x_2)$ over $x_2$.

*Affine Transformation of Coordinates*

Let

$$g(x_1, x_2) = s(ax_1 + bx_2 + c,\ dx_1 + ex_2 + f)$$

denote an affine transformation of coordinates. If $S(u_1, u_2)$ is the 2D Fourier transform of $s(x_1, x_2)$, then the 2D Fourier transform of $g(x_1, x_2)$ is given by

$$G(u_1, u_2) = \frac{1}{ae - bd}\, e^{j2\pi \frac{[(ec-bf)u_1 + (af-cd)u_2]}{ae-bd}}\, S\left(\frac{eu_1 - du_2}{ae - bd}, \frac{-bu_1 + au_2}{ae - bd}\right) \tag{1.7a}$$

*Proof*

The Fourier transform is given by

$$G(u_1, u_2) = \iint s(ax_1 + bx_2 + c, dx_1 + ex_2 + f)\, e^{-j2\pi(u_1x_1 + u_2x_2)}\, dx_1\, dx_2$$

In order to perform a change of variables, we can first write the affine mapping in vector-matrix notation as

$$\begin{bmatrix} x_1' \\ x_2' \end{bmatrix} = \begin{bmatrix} a & b \\ d & e \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} c \\ f \end{bmatrix}$$

and the Jacobian relation as $dx_1' dx_2' = |\Delta|\, dx_1 dx_2$, where $\Delta = ae - bd$. Provided that $\Delta \neq 0$, the inverse relation is given by

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} a & b \\ d & e \end{bmatrix}^{-1} \begin{bmatrix} x_1' - c \\ x_2' - f \end{bmatrix}$$

Then,

$$u_1x_1 + u_2x_2 = \begin{bmatrix} u_1 & u_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$= \begin{bmatrix} u_1 & u_2 \end{bmatrix} \begin{bmatrix} a & b \\ d & e \end{bmatrix}^{-1} \begin{bmatrix} x_1' \\ x_2' \end{bmatrix} - \begin{bmatrix} u_1 & u_2 \end{bmatrix} \begin{bmatrix} a & b \\ d & e \end{bmatrix}^{-1} \begin{bmatrix} c \\ f \end{bmatrix}$$

and

$$G(u_1,u_2)=\iint s(x_1',x_2')\, e^{j2\pi\frac{[(ec-bf)u_1+(af-cd)u_2]}{ae-bd}}\, e^{j2\pi\frac{[(eu_1-du_2)x_1'+(-bu_1+au_2)x_2']}{ae-bd}}\, \frac{dx_1'\,dx_2'}{ae-bd}$$

$$=\frac{1}{ae-bd}\, e^{j2\pi\frac{[(ec-bf)u_1+(af-cd)u_2]}{ae-bd}}\, S\left(\frac{eu_1-du_2}{ae-bd},\frac{-bu_1+au_2}{ae-bd}\right)$$

### Special Cases

1. *Translation:* $a = e = 1,\ b = d = 0.$

$$G(u_1,u_2)=e^{j2\pi(cu_1+fu_2)}S(u_1,u_2) \tag{1.7b}$$

   which states that translation in the spatial coordinates results in a phase-shift in the Fourier domain as expected.
2. *Rotation:* $c=f=0,\ a=e=\cos\theta,\ d=-b=\sin\theta$, where $ae - bd = 1$.

$$G(u_1,u_2)=S(u_1\cos\theta-u_2\sin\theta,\ u_1\sin\theta+u_2\cos\theta) \tag{1.7c}$$

   which states that a rotation of spatial coordinates results in a rotation of Fourier coordinates by an equal angle. Hence, the Fourier transform is rotation invariant.

### Projection Slice Theorem

Let $s(x_1,x_2)$ have Fourier transform $S(u_1,u_2)$, and $p_\theta(x_1)$ denote the Radon transform of $s(x_1,x_2)$ defined by

$$p_\theta(x_1)=\int_{-\infty}^{\infty} s(x_1\cos\theta+x_2\sin\theta,-x_1\sin\theta+x_2\cos\theta)\,dx_2 \tag{1.8a}$$

which projects $s(x_1,x_2)$ to a line through the origin with angle $\theta$. Then,

$$P_\theta(\Omega)=S(\Omega\cos\theta,\Omega\sin\theta) \tag{1.8b}$$

where $P_\theta(\Omega)$ denotes the 1D Fourier transform of $p_\theta(x_1)$ for each angle $\theta$.

In words, the projection slice theorem says that taking a function $s(x_1,x_2)$ and first projecting it to a line through the origin with angle $\theta$ and then taking its 1D Fourier transform is equivalent to taking the 2D Fourier transform of $s(x_1,x_2)$ and

evaluating the Fourier transform $S(u_1, u_2)$ on the line through the origin with angle $\theta$. This can be easily proven for the case $\theta = 0$. Projection of $s(x_1, x_2)$ on the horizontal axis is defined by

$$p_0(x_1) = \int_{-\infty}^{\infty} s(x_1, x_2)\, dx_2$$

Taking the 1D Fourier transform of the projection $p_0(x_1)$ yields

$$P_0(u_1) = \int_{-\infty}^{\infty} \left\{ \int_{-\infty}^{\infty} s(x_1, x_2)\, dx_2 \right\} e^{-ju_1x_1}\, dx_1$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} s(x_1, x_2) e^{-ju_1x_1}\, dx_1 dx_2 = S(u_1, 0)$$

which provides the desired result. The projection-slice theorem is fundamental to how several medical imaging modalities (e.g., computer tomography) work.

## 1.2.2 Fourier Transform of Discrete Signals

Many properties of the MD Fourier transform of continuous signals also hold for MD Fourier transform of discrete signals. However, there are also some important differences. Most important among these is that the Fourier transform of MD discrete signals is rectangularly periodic.

**Definition:** The Fourier transform of a 2D discrete signal $s(n_1, n_2)$ is a rectangularly periodic function of "normalized" continuous frequency variables $\omega_1$ and $\omega_2$, where

$$S\left(e^{j\omega_1}, e^{j\omega_2}\right) = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} s(n_1, n_2) e^{-j(\omega_1 n_1 + \omega_2 n_2)} \tag{1.9a}$$

is periodic with period $2\pi \times 2\pi$, and the inverse 2D transform is given by

$$s(n_1, n_2) = \frac{1}{(2\pi)^2} \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} S(e^{j\omega_1}, e^{j\omega_2}) e^{j(\omega_1 n_1 + \omega_2 n_2)}\, d\omega_1\, d\omega_2 \tag{1.9b}$$

The same concepts of uniform convergence, mean-square convergence, and generalized convergence that have been discussed in Section 1.2.1 also apply here, but this time in the context of convergence of infinite series rather than improper integrals.

**Properties of the 2D Fourier Transform of Discrete Signals**

1. The Fourier transform $S\left(e^{j\omega_1}, e^{j\omega_2}\right)$ is periodic with the period $2\pi$ in both $\omega_1$ and $\omega_2$:

$$S\left(e^{j\omega_1}, e^{j\omega_2}\right) = S\left(e^{j(\omega_1+2\pi)}, e^{j\omega_2}\right) = S\left(e^{j\omega_1}, e^{j(\omega_2+2\pi)}\right) \tag{1.10}$$

Therefore, $S\left(e^{j\omega_1}, e^{j\omega_2}\right)$ is usually specified in the unit cell $[-\pi, \pi) \times [-\pi, \pi)$.

2. The Fourier transform is complex:

$$\begin{aligned} S\left(e^{j\omega_1}, e^{j\omega_2}\right) &= \left|S\left(e^{j\omega_1}, e^{j\omega_2}\right)\right| e^{j\theta_S(\omega_1,\omega_2)} \\ &= S_R\left(e^{j\omega_1}, e^{j\omega_2}\right) + jS_I\left(e^{j\omega_1}, e^{j\omega_2}\right) \end{aligned}$$

where $S_R\left(e^{j\omega_1}, e^{j\omega_2}\right)$ and $S_I\left(e^{j\omega_1}, e^{j\omega_2}\right)$ denote the real and imaginary parts, respectively, $\left|S\left(e^{j\omega_1}, e^{j\omega_2}\right)\right|$ denotes the Fourier magnitude, and $\theta_S(\omega_1,\omega_2)$ denotes the Fourier phase. Note that the imaginary part $S_I\left(e^{j\omega_1}, e^{j\omega_2}\right)$ is real.

3. Given a 2D complex signal $s(n_1, n_2)$, we can define its conjugate symmetric part $s_e(n_1, n_2)$ and conjugate anti-symmetric part $s_o(n_1, n_2)$ as

$$s_e(n_1,n_2) = \tfrac{1}{2}\{s(n_1,n_2) + s^*(-n_1,-n_2)\} \tag{1.11a}$$

$$s_o(n_1,n_2) = \tfrac{1}{2}\{s(n_1,n_2) - s^*(-n_1,-n_2)\} \tag{1.11b}$$

Then, we have the following Fourier transform pairs:

$$s_e(n_1,n_2) \leftrightarrow S_R\left(e^{j\omega_1}, e^{j\omega_2}\right)$$

$$s_o(n_1,n_2) \leftrightarrow jS_I\left(e^{j\omega_1}, e^{j\omega_2}\right)$$

4. If $s(n_1, n_2)$ is real, then $S\left(e^{j\omega_1}, e^{j\omega_2}\right)$ is Hermitian symmetric, i.e.,

$$S\left(e^{j\omega_1}, e^{j\omega_2}\right) = S^*\left(e^{-j\omega_1}, e^{-j\omega_2}\right) \tag{1.12}$$

which implies that the magnitude is an even function and the phase is an odd function, i.e., $\left|S\left(e^{j\omega_1}, e^{j\omega_2}\right)\right| = \left|S\left(e^{-j\omega_1}, e^{-j\omega_2}\right)\right|$ and $\theta_S(\omega_1,\omega_2) = -\theta_S(-\omega_1,-\omega_2)$.

It also implies that the real part is even and the imaginary part is odd, i.e., $S_R\left(e^{j\omega_1}, e^{j\omega_2}\right) = S_R\left(e^{-j\omega_1}, e^{-j\omega_2}\right)$ and $S_I\left(e^{j\omega_1}, e^{j\omega_2}\right) = -S_I\left(e^{-j\omega_1}, e^{-j\omega_2}\right)$.

Since we have two-fold symmetry, it is sufficient to specify the value of $S\left(e^{j\omega_1}, e^{j\omega_2}\right)$ over the NSHP support depicted in Figure 1.1(c).

5. Spatial-domain symmetry and zero Fourier-phase: A signal $s(n_1, n_2)$ has zero Fourier-phase if $S\left(e^{j\omega_1}, e^{j\omega_2}\right)$ is real and positive. If $S\left(e^{j\omega_1}, e^{j\omega_2}\right)$ is real but negative for some $(\omega_1, \omega_2)$, then it has a phase of 180 degrees (or $\pi$ radians) for those $(\omega_1, \omega_2)$ where it is negative. In either case, $S\left(e^{j\omega_1}, e^{j\omega_2}\right) = S^*\left(e^{j\omega_1}, e^{j\omega_2}\right)$ implies that $s(n_1, n_2) = s^*(-n_1, -n_2)$, which is known as two-fold spatial symmetry. A stronger symmetry is four-fold symmetry, which (for real-valued signals) can be stated as $s(n_1, n_2) = s(-n_1, n_2) = s(n_1, -n_2)$. In the Fourier domain, four-fold symmetry implies $S\left(e^{j\omega_1}, e^{j\omega_2}\right) = S\left(e^{-j\omega_1}, e^{j\omega_2}\right) = S\left(e^{j\omega_1}, e^{-j\omega_2}\right)$.

### 1.2.3 Discrete Fourier Transform (DFT)

The Fourier transform $S\left(e^{j\omega_1}, e^{j\omega_2}\right)$ of a 2D discrete signal is a function of two continuous variables $-\pi \le \omega_1 < \pi$ and $-\pi \le \omega_2 < \pi$. For a digital representation, the frequency variables must be discretized or sampled. It turns out that this is only possible for finite-extent discrete signals. Since finite-extent signals and periodic signals are isomorphic to each other, the DFT of $s(n_1, n_2)$ with $N_1 \times N_2$ samples is equal to the discrete Fourier series coefficients of the periodically extended signal $\tilde{s}(n_1, n_2)$ with period at least $N_1 \times N_2$. The Fourier series coefficients themselves are periodic with period $N_1 \times N_2$. The main period of the Fourier series coefficients is defined as the DFT of the finite-extent signal, given by

$$S(k_1, k_2) = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} \tilde{s}(n_1, n_2) e^{-j\left(\frac{2\pi k_1}{N_1} n_1 + \frac{2\pi k_2}{N_2} n_2\right)},$$
$$0 \le k_1 < N_1 - 1,\ 0 \le k_2 < N_2 - 1 \tag{1.13}$$

Alternatively, it is possible to obtain the DFT $S(k_1, k_2)$ by sampling the Fourier transform $S\left(e^{j\omega_1}, e^{j\omega_2}\right)$ with $N_1 \times N_2$ samples over the unit cell $[0, 2\pi) \times [0, 2\pi)$. Given that $s(n_1, n_2)$ is finite extent with at most $N_1 \times N_2$ samples, there will be no spatial-domain aliasing due to sampling of the Fourier transform $S\left(e^{j\omega_1}, e^{j\omega_2}\right)$. On the other hand, if we sample the Fourier transform of an infinite-extent sequence $s(n_1, n_2)$, or if $s(n_1, n_2)$ has more than $N_1 \times N_2$ samples, then $\tilde{s}(n_1, n_2)$, obtained by computing the $N_1 \times N_2$ point inverse DFT of these samples $S(k_1, k_2)$, will exhibit space-domain aliasing, where $\tilde{s}(n_1, n_2)$ is related to $s(n_1, n_2)$ by

$$\tilde{s}(n_1,n_2)=\sum_{i_1=-\infty}^{\infty}\sum_{i_2=-\infty}^{\infty}s(n_1-i_1N_1,\, n_2-i_2N_2)$$

**Convergence**

There is no issue of convergence with the DFT, since Eqn. (1.13) involves finite summations only.

**Normalized Frequency Variables**

The variables $(k_1, k_2)$ are unitless indices, while the unit of the variables $(\omega_1,\omega_2)$ is radians. They are related to the physical frequency variables $(u_1, u_2)$ whose unit is cycles/distance by

$$u_1=\frac{\omega_1}{\Delta x_1}=\frac{2\pi k_1}{N_1\Delta x_1} \text{ and } u_2=\frac{\omega_2}{\Delta x_2}=\frac{2\pi k_2}{N_2\Delta x_2} \tag{1.14}$$

where $\Delta x_1$ and $\Delta x_2$ are sampling distances in the horizontal and vertical dimensions in the spatial coordinates.

**Computation of the 2D DFT and Inverse 2D DFT**

Since 2D complex exponentials are separable, 2D DFT can be computed as a cascade of two 1D DFTs, first on the rows of $s(n_1, n_2)$ then on the columns of $S(n_1, k_2)$ as

$$S(k_1,k_2)=\sum_{n_1=0}^{N_1-1}\left[\sum_{n_2=0}^{N_2-1}s(n_1,n_2)e^{-j\frac{2\pi k_2}{N_2}n_2}\right]e^{-j\frac{2\pi k_1}{N_1}n_1}$$

where

$$S(n_1,k_2)=\sum_{n_2=0}^{N_2-1}s(n_1,n_2)e^{-j\frac{2\pi k_2}{N_2}n_2}$$

is the 1D DFT over the row $n_2$ of the image. Note that 1D DFTs are computed by using the Fast Fourier Transform (FFT) algorithm.

Inverse 2D DFT can be computed using the forward FFT algorithm, as illustrated in Figure 1.4, by first conjugating $S(k_1, k_2)$, then computing 2D forward DFT, and finally again taking the complex conjugate of the result, since we have

$$\begin{aligned}s(n_1,n_2)&=\frac{1}{N_1N_2}\sum_{k_1=0}^{N_1-1}\sum_{k_2=0}^{N_2-1}S(k_1,k_2)e^{j\left(\frac{2\pi k_1}{N_1}n_1+\frac{2\pi k_2}{N_2}n_2\right)}\\&=\frac{1}{N_1N_2}\sum_{k_1=0}^{N_1-1}\sum_{k_2=0}^{N_2-1}\left(S^*(k_1,k_2)e^{-j\left(\frac{2\pi k_1}{N_1}n_1+\frac{2\pi k_2}{N_2}n_2\right)}\right)^*\end{aligned}$$




**Figure 1.4** Computation of 2D inverse DFT using FFT algorithms.

## Properties of the DFT

1. The 2D DFT is rectangularly periodic with period $N_1 \times N_2$, i.e.,

$$S(k_1,k_2) = S(k_1 + N_1, k_2) = S(k_1, k_2 + N_2) \text{ for all } (k_1,k_2)$$

The 2D DFT $S(k_1,k_2)$ is complex valued and can be expressed as

$$\begin{aligned} S(k_1,k_2) &= |S(k_1,k_2)|\, e^{j\Phi(k_1,k_2)} \\ &= S_R(k_1,k_2) + jS_I(k_1,k_2) \end{aligned}$$

where $|S(k_1,k_2)|$ and $\Phi(k_1,k_2)$ denote the magnitude and phase of the DFT, and $S_R(k_1,k_2)$ and $S_I(k_1,k_2)$ are the real and imaginary parts, respectively.

2. If we define the conjugate symmetric part $s_e(n_1,n_2)$ and conjugate anti-symmetric part $s_o(n_1,n_2)$ of a complex signal $s(n_1,n_2)$ as in (1.11), then we have the following Fourier transform pairs:

$$\begin{aligned} s_e(n_1,n_2) &\leftrightarrow S_R(k_1,\ k_2) \\ s_o(n_1,n_2) &\leftrightarrow jS_I(k_1,\ k_2) \end{aligned}$$

3. For real-valued signals $s(n_1,n_2)$, the DFT is Hermitian symmetric, which means

$$S(k_1,k_2) = S^*(-k_1,-k_2)$$

or equivalently $|S(k_1,\ k_2)| = |S(-k_1,-k_2)|$ and $\Phi(k_1,\ k_2) = -\Phi(-k_1,\ -k_2)$.

Because of this symmetry, the total number of non-redundant DFT magnitude and phase samples is equal to the number of image samples. An example for $N_1 = N_2 = 6$ is shown in Figure 1.5. Note that $\Phi(0,0) = -\Phi(0,0) = 0$; $\Phi(3,0) = -\Phi(-3,0) = -\Phi(-3+6,0) = 0$; $\Phi(0,3) = -\Phi(0,-3+6) = -\Phi(0,3) = 0$; and $\Phi(3,3) = -\Phi(-3,-3) = 0$ because of the periodicity of the DFT. There are only 20 distinct DFT magnitude coefficients and 16

| | | | | | |
|---|---|---|---|---|---|
| $\|S(0,0)\|$ | $\|S(1,0)\|$ | $\|S(2,0)\|$ | $\|S(3,0)\|$ | $\|S(2,0)\|$ | $\|S(1,0)\|$ |
| $\|S(0,1)\|$ | $\|S(1,1)\|$ | $\|S(2,1)\|$ | $\|S(3,1)\|$ | $\|S(4,1)\|$ | $\|S(5,1)\|$ |
| $\|S(0,2)\|$ | $\|S(1,2)\|$ | $\|S(2,2)\|$ | $\|S(3,2)\|$ | $\|S(4,2)\|$ | $\|S(5,2)\|$ |
| $\|S(0,3)\|$ | $\|S(1,3)\|$ | $\|S(2,3)\|$ | $\|S(3,3)\|$ | $\|S(2,3)\|$ | $\|S(1,3)\|$ |
| $\|S(0,2)\|$ | $\|S(5,2)\|$ | $\|S(4,2)\|$ | $\|S(3,2)\|$ | $\|S(2,2)\|$ | $\|S(1,2)\|$ |
| $\|S(0,1)\|$ | $\|S(5,1)\|$ | $\|S(4,1)\|$ | $\|S(3,1)\|$ | $\|S(2,1)\|$ | $\|S(1,1)\|$ |

(a)

| | | | | | |
|---|---|---|---|---|---|
| 0 | $\Phi(1,0)$ | $\Phi(2,0)$ | 0 | $-\Phi(2,0)$ | $-\Phi(1,0)$ |
| $\Phi(0,1)$ | $\Phi(1,1)$ | $\Phi(2,1)$ | $\Phi(3,1)$ | $\Phi(4,1)$ | $\Phi(5,1)$ |
| $\Phi(0,2)$ | $\Phi(1,2)$ | $\Phi(2,2)$ | $\Phi(3,2)$ | $\Phi(4,2)$ | $\Phi(5,2)$ |
| 0 | $\Phi(1,3)$ | $\Phi(2,3)$ | 0 | $-\Phi(2,3)$ | $-\Phi(1,3)$ |
| $-\Phi(0,2)$ | $-\Phi(5,2)$ | $-\Phi(4,2)$ | $-\Phi(3,2)$ | $-\Phi(2,2)$ | $-\Phi(1,2)$ |
| $-\Phi(0,1)$ | $-\Phi(5,1)$ | $-\Phi(4,1)$ | $-\Phi(3,1)$ | $-\Phi(2,1)$ | $-\Phi(1,1)$ |

(b)

**Figure 1.5** Hermitian symmetry of $S(k_1, k_2)$ for real $(n_1, n_2)$. Gray-shaded coefficients are distinct, and others are determined by the symmetry. (a) The magnitude is even; (b) the phase is odd.

distinct DFT phase coefficients, for a total of $6 \times 6 = 36$ distinct DFT coefficients for a 36-pixel image.

4. Circular Shift: Since $s(n_1, n_2)$ is $N_1 \times N_2$ finite-extent signal, any shift of coordinates must be such that the shifted signal must remain within the original $N_1 \times N_2$ support. This is called a circular shift and can be defined by using the modulo operation $\langle \cdot \rangle_N$. We have the following Fourier transform relation:

$$s\left(\langle n_1 - M_1 \rangle_{N_1}, \langle n_2 - M_2 \rangle_{N_2}\right) \leftrightarrow S(k_1, k_2) e^{-j2\pi\left(\frac{M_1 k_1}{N_1} + \frac{M_2 k_2}{N_2}\right)}$$

5. Shift of Origin: It is usually desirable to shift the origin to the center of the image to obtain more visually pleasing 2D plots. This can be achieved by multiplying the image

$$s_0(n_1, n_2) = (-1)^{n_1 + n_2} s(n_1, n_2) \tag{1.15}$$

prior to computing the DFT. Using the frequency shifting property of the Fourier transform

$$e^{j\frac{2\pi}{N_1}\frac{N_1}{2}n_1} e^{j\frac{2\pi}{N_2}\frac{N_2}{2}n_2} s(n_1, n_2) \leftrightarrow S\left(k_1 - \frac{N_1}{2}, k_2 - \frac{N_2}{2}\right)$$

since

$$e^{j\frac{2\pi}{N_1}\frac{N_1}{2}n_1} e^{j\frac{2\pi}{N_2}\frac{N_2}{2}n_2} = e^{j\pi(n_1+n_2)} = (-1)^{(n_1+n_2)}$$

6. Parseval's Theorem: DFT preserves energy; i.e., the energy in the spatio-temporal domain is unchanged in the Fourier domain, which can be expressed as

$$\sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} \left|s(n_1, n_2)\right|^2 = \frac{1}{N_1 N_2} \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} \left|S(k_1, k_2)\right|^2 \tag{1.16}$$

### 1.2.4 Discrete Cosine Transform (DCT)

The DCT (or its integer approximations) is often the transform of choice in the state-of-the-art image and video-compression standards for the following reasons: i) Unlike the DFT, it is real valued for real-valued images (note that the DCT is not the same as the real part of the DFT); ii) it can be computed by FFT algorithms (developed to compute the DFT) via a symmetric extension of the signal; and iii) the high-frequency coefficients of the DCT contains less energy compared with those of the DFT, since intensity discontinuity at the image boundaries due to implicit periodic extension in DFT is alleviated by the symmetric extension in DCT.

Similar to 2D-DFT, 2D-DCT basis functions are separable; hence, the computation of 2D-DCT and inverse DCT can be performed as a cascade of two 1D-DCTs just as in the 2D-DFT. That is, we first compute the 1D-DCT of columns of the image, followed by the 1D-DCT of rows of the intermediate result.

There are eight types of 1D-DCT depending on the method used for the symmetric extension. DCT types I–IV treat right and left boundaries consistently regarding the point of symmetry; i.e., they are even/odd around either a data point or halfway between two data points for both boundaries. On the other hand, in DCT types V–VIII, the symmetry type alternates between right and left boundaries; i.e., it is even/odd around a data point for one boundary and halfway between two data points for the other boundary.

The most common form of DCT is the type II DCT where both boundaries are symmetric with respect to halfway between two samples; i.e., the last sample repeats. Type II 2D-DCT is given by

$$C(k_1,k_2)=\sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1} 4\, s(n_1,n_2)\cos\left(\frac{\pi k_1}{2N_1}(2n_1+1)\right)\cos\left(\frac{\pi k_2}{2N_2}(2n_2+1)\right)$$

$$0 \le k_1 < N_1 - 1,\ 0 \le k_2 < N_2 - 1 \tag{1.17}$$

and the inverse 2D DCT is given by

$$s(n_1,n_2)=\frac{1}{N_1N_2}\sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1} w(k_1)w(k_2)\ C(k_1,k_2)$$

$$\cos\left(\frac{\pi k_1}{2N_1}(2n_1+1)\right)\ \cos\left(\frac{\pi k_2}{2N_2}(2n_2+1)\right)$$

where

$$w(k)=\begin{cases} 1/2 & k=0 \\ 1 & k\neq 0 \end{cases}$$

**Relationship to the DFT**

Each $N$-point 1D-DCT over the columns (rows) can be computed by a $2N$-point 1D-FFT after symmetric extension over the columns (rows). Symmetric extension for type II DCT is given by

$$g(n) = s(n) + s(2N - 1 - n),\ 0 \le n \le 2N - 1$$

where $s(n)$ is an $N$-point signal; i.e., $s(n) = 0$, $N \le n \le 2N - 1$. Note that $g(n)$ is symmetric with respect to sample $N - 1/2$, which is between the last original sample $N - 1$ and the first extended sample $N$, and $g(N) = s(N - 1)$ and $g(2N - 1) = s(0)$. The algorithm to compute the $N$-point DCT using the $2N$-point FFT is as follows:

1. Form the $2N$-point symmetrically extended signal $g(n)$.
2. Compute $G(k)$, $k = 0, \ldots, 2N - 1$ the $2N$-point DFT of $g(n)$.
3. The $N$-point DCT $C(k) = W_{2N}^{k/2}G(k)$, $k = 0, \ldots, N-1$, where $W_{2N}^{k} = e^{-j\frac{2\pi k}{2N}}$.

In the case of 2D-DCT, the 2D-DCT of an $N_1 \times N_2$ image is related to the first quadrant ($N_1 \times N_2$ points) of a $2N_1 \times 2N_2$ point 2D-DFT of the symmetrically extended signal. Symmetric extension must be performed both along the columns (before the column DCT) and along the rows (before the row DCT):

$$g(n_1, n_2) = s(n_1, n_2) + s(n_1, 2N_2 - 1 - n_2) + s(2N_1 - 1 - n_1, n_2) \\ + s(2N_1 - 1 - n_1, 2N_2 - 1 - n_2)$$

Hence, the resulting symmetrically extended 2D signal exhibits four-fold symmetry.

Finally, the $N_1 \times N_2$ point DCT coefficients $C(k_1, k_2)$ can be expressed in terms of $2N_1 \times 2N_2$ point 2D-DFT values $G(k_1, k_2)$ by

$$C(k_1, k_2) = e^{-j\pi\left(\frac{k_1}{2N_1} + \frac{k_2}{2N_2}\right)} G(k_1, k_2), \text{ for } (k_1, k_2) \in [0, N_1 - 1] \times [0, N_2 - 1]$$

Note that the phase factor offsets the phase of the DFT that originates from the half-point symmetry about the point $\left(N_1 - 1/2, N_2 - 1/2\right)$.

## 1.3 Multi-Dimensional Systems

Classical signal-processing texts study mainly linear shift-invariant (LSI) systems (filters), which can be classified as finite-impulse response (FIR) and infinite-impulse response (IIR) filters [Lim 90]. However, most image- and video-processing applications require directional and/or adaptive filters, which are not LSI systems. Edge-adaptive filters in image-processing and motion-compensated filters for video processing are examples of such directional filtering. Nevertheless, this section focuses on LSI filters since LSI-FIR filters are used in some important applications, and it is important to gain an understanding of the limitations of LSI filters. Adaptive filters are covered in subsequent chapters in the context of specific applications. We do not review the definitions of linearity and shift-invariance here as they can be found elsewhere.

### 1.3.1 Impulse Response and 2D Convolution

Just as in 1D systems, MD-LSI filters can be completely specified by their impulse response. In 2D, impulse response is the response of a 2D system $\boldsymbol{L}$ to a 2D unit impulse input, denoted by

$$h(n_1, n_2) = \boldsymbol{L}[\delta(n_1, n_2)] \tag{1.18}$$

The output of a 2D linear shift-invariant system to an arbitrary input $s(n_1, n_2)$ can be expressed as

$$g(n_1,n_2) = \boldsymbol{L}\left[s(n_1,n_2)\right] \tag{1.19}$$

Expressing the input as a sum of weighted and shifted 2D unit impulses, as stated by Eqn. (1.5b), and substituting (1.5b) into (1.19)

$$g(n_1,n_2) = \boldsymbol{L}\left[\sum_{k_1}\sum_{k_2} s(k_1,k_2)\,\delta(n_1-k_1,n_2-k_2)\right]$$

Changing the order of linear operator **L** and double summation, using linearity

$$g(n_1,n_2) = \sum_{k_1}\sum_{k_2} s(k_1,k_2)\,\boldsymbol{L}\left[\delta(n_1-k_1,n_2-k_2)\right]$$

Now using the definition of the impulse response together with shift-invariance of $\boldsymbol{L}$, we obtain

$$g(n_1,n_2) = \textstyle\sum_{k_1}\sum_{k_2} s(k_1,k_2)h(n_1-k_1,n_2-k_2) \tag{1.20a}$$

which, by a change of variables, is equivalent to

$$g(n_1,n_2) = \textstyle\sum_{k_1}\sum_{k_2} h(k_1,k_2)s(n_1-k_1,n_2-k_2) \tag{1.20b}$$

Both (1.20a) and (1.20b) are known as 2D convolution summation. In theory, both $s(n_1,n_2)$ and $h(n_1,n_2)$ may have infinite support, which results in infinite summations. Then, it can be shown that the convolution summation converges if

$$\textstyle\sum_{n_1=-\infty}^{\infty}\sum_{n_2=-\infty}^{\infty}\left|h(n_1,n_2)\right| < \infty \tag{1.21}$$

### Stable Filters

A 2D-LTI system is called stable if and only if its impulse response satisfies Eqn. (1.21). The impulse response of all FIR filters satisfies (1.21); hence, all FIR filters are stable. There are tests to check stability of IIR filters [Wds 06]. Only stable IIR filters can be implemented.

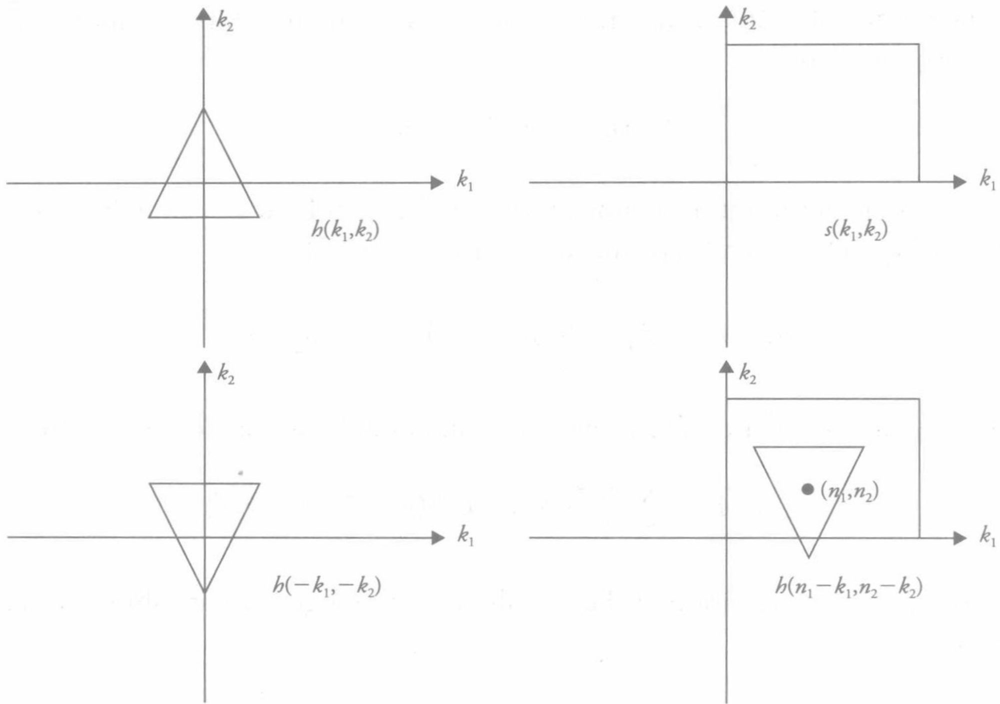



**Figure 1.6** Graphical illustration of 2D convolution.

### Computation of the Convolution Summation

Computation of 2D convolution summation (1.20) is illustrated in Figure 1.6. We first flip $h(k_1, k_2)$ both along the $k_1$ and $k_2$ axes to obtain $h(-k_1, -k_2)$, which is then shifted over $s(k_1, k_2)$ for specific values of $n_1$ and $n_2$. We repeat this procedure for all possible values of $n_1$ and $n_2$.

Numerical computation of (1.20) is possible if both the input image and the impulse response of the filter are finite extent. If we assume the input image is $N_1 \times N_2$, and the support of the filter $h(n_1, n_2)$ is $M_1 \times M_2$, then the output will be $(N_1+M_1-1) \times (N_2+M_2-1)$ points. Then, the implementation of 2D convolution summation (1.20) requires $M_1M_2$ multiplies and $M_1M_2$ adds per pixel, which may be time consuming if $M_1$ and $M_2$ are large.

### Separable Filters

Convolution using separable kernels results in significant computational savings. A filter is called separable if its impulse response is separable.

$$h(n_1, n_2) = h_1(n_1)h_2(n_2)$$

Convolution with a 2D separable impulse response can be written as a cascade of two 1D convolutions:

$$
\begin{aligned}
g(n_1, n_2) &= s(n_1, n_2) ** h(n_1, n_2) \\
&= \sum_{k_1}\sum_{k_2} s(k_1, k_2)\; h(n_1 - k_1, n_2 - k_2) \\
&= \sum_{k_1}\sum_{k_2} s(k_1, k_2)\, h_1(n_1 - k_1)\, h_2(n_2 - k_2) \\
&= \sum_{k_1} h_1(n_1 - k_1) \sum_{k_2} s(k_1, k_2)\; h_2(n_2 - k_2) \\
&= \sum_{k_1} h_1(n_1 - k_1)\, y(k_1, n_2) \\
&= h_1(n_1) * \left[h_2(n_2) * s(n_1, n_2)\right]
\end{aligned}
$$

Therefore, implementation of the separable convolution requires $M$ multiplies and $M$ adds per pixel for each 1D convolution, hence, a total of $2M$ multiplies and $2M$ additions. For a typical $15 \times 15$ filter, this means 30 multiplies and adds instead of 225 multiplies and adds per pixel.

## 1.3.2 Frequency Response

Stable LSI systems can also be characterized by their frequency response. The frequency response of an LSI system is defined in terms of its output $g(n_1, n_2)$ to a complex exponential input

$$
s(n_1, n_2) = e^{j(\omega_1 n_1 + \omega_2 n_2)} \tag{1.22}
$$

Substituting (1.22) into (1.20), we obtain

$$
g(n_1, n_2) = \sum_{k_1}\sum_{k_2} h(k_1, k_2)\, e^{j[\omega_1(n_1 - k_1) + \omega_2(n_2 - k_2)]}
$$

Taking the terms that do not depend on $k_1$ and $k_2$ out of the summations,

$$
\begin{aligned}
g(n_1, n_2) &= e^{j(\omega_1 n_1 + \omega_2 n_2)} \sum_{k_1}\sum_{k_2} h(k_1, k_2)\, e^{-j(\omega_1 k_1 + \omega_2 k_2)} \\
&= e^{j(\omega_1 n_1 + \omega_2 n_2)} H\left(e^{j\omega_1}, e^{j\omega_2}\right)
\end{aligned}
$$

where

$$
H\left(e^{j\omega_1}, e^{j\omega_2}\right) = \sum_{k_1}\sum_{k_2} h(k_1, k_2) e^{-j(\omega_1 k_1 + \omega_2 k_2)} \tag{1.23}
$$

is called the frequency response of the system. We observe that the output of an LSI system to a complex exponential input is the same input multiplied by a complex-valued "frequency response," which is a function of frequencies $\omega_1$ and $\omega_2$. Note that the frequency response is the MD Fourier transform of the system impulse response.

The frequency response of separable filters is also separable and can be written as

$$H\left(e^{j\omega_1}, e^{j\omega_2}\right) = H_1\left(e^{j\omega_1}\right) H_2\left(e^{j\omega_2}\right) \tag{1.24}$$

### Magnitude and Phase of the Frequency Response

Since the frequency response is a complex-valued function, it can be expressed in terms of its real part $H_R\left(e^{j\omega_1}, e^{j\omega_2}\right)$ and imaginary part $H_I\left(e^{j\omega_1}, e^{j\omega_2}\right)$, which are both real-valued functions:

$$H\left(e^{j\omega_1}, e^{j\omega_2}\right) = H_R\left(e^{j\omega_1}, e^{j\omega_2}\right) + jH_I\left(e^{j\omega_1}, e^{j\omega_2}\right) \tag{1.25a}$$

or in terms of a real, positive magnitude and a real phase function $\theta(\omega_1, \omega_2)$

$$H\left(e^{j\omega_1}, e^{j\omega_2}\right) = \left|H\left(e^{j\omega_1}, e^{j\omega_2}\right)\right| e^{j\theta\left(e^{\omega_1}, e^{\omega_2}\right)} \tag{1.25b}$$

where

$$\left|H\left(e^{j\omega_1}, e^{j\omega_2}\right)\right| = \sqrt{H_R^2\left(e^{j\omega_1}, e^{j\omega_2}\right) + H_I^2\left(e^{j\omega_1}, e^{j\omega_2}\right)} \tag{1.26a}$$

and

$$\theta\left(e^{\omega_1}, e^{\omega_2}\right) = \tan^{-1}\left(\frac{H_I\left(e^{j\omega_1}, e^{j\omega_2}\right)}{H_R\left(e^{j\omega_1}, e^{j\omega_2}\right)}\right) \tag{1.26b}$$

The phase response of a filter plays a vital role in image processing. Phase distortions introduced by filtering are visible to the eye as artifacts. Therefore, it is highly desirable that filters used in image processing have zero or linear phase. This can be achieved by using FIR filters with symmetry properties as discussed in Section 1.3.3.

### Convolution in the Fourier Domain

The output of an LSI filter to an arbitrary input can also be computed in the Fourier domain using the convolution property of the Fourier transform. If we take the Fourier transform of the convolution summation

$$G\left(e^{j\omega_1}, e^{j\omega_2}\right) = \sum_{n_1=-\infty}^{\infty}\sum_{n_2=-\infty}^{\infty}\left(\sum_{k_1}\sum_{k_2} h(k_1,k_2)s\left(n_1-k_1,n_2-k_2\right)\right)e^{-j(\omega_1 n_1+\omega_2 n_2)}$$

Rewriting the exponential function

$$G\left(e^{j\omega_1}, e^{j\omega_2}\right) = \sum_{n_1}\sum_{n_2}\left(\sum_{k_1}\sum_{k_2} h(k_1,k_2)s(n_1-k_1,n_2-k_2)\right)e^{-j\{\omega_1(k_1+(n_1-k_1))+\omega_2(k_2+(n_2-k_2))\}}$$

and rearranging the terms

$$G\left(e^{j\omega_1}, e^{j\omega_2}\right) = \sum_{k_1}\sum_{k_2} h(k_1,k_2)e^{-j(\omega_1 k_1+\omega_2 k_2)}\sum_{n_1}\sum_{n_2} s(n_1-k_1,n_2-k_2)e^{-j\{\omega_1(n_1-k_1)+\omega_2(n_2-k_2)\}}$$

if we let $n_1' = n_1 - k_1$ and $n_2' = n_2 - k_2$, we obtain

$$\begin{aligned} G\left(e^{j\omega_1}, e^{j\omega_2}\right) &= \left(\sum_{k_1}\sum_{k_2} h(k_1,k_2)e^{-j(\omega_1 k_1+\omega_2 k_2)}\right)\left(\sum_{n_1}^{\infty}\sum_{n_1}^{\infty} s\left(n_1',n_2'\right)e^{-j(\omega_1 n_1'+\omega_2 n_2')}\right) \\ &= H\left(e^{j\omega_1}, e^{j\omega_2}\right)S\left(e^{j\omega_1}, e^{j\omega_2}\right) \end{aligned} \tag{1.27}$$

Therefore, the Fourier transform of the output at the frequency $(\omega_1, \omega_2)$ depends on the Fourier transform of the input and the frequency response of the system only at the frequency $(\omega_1, \omega_2)$, and a stable LSI system can be completely specified by its impulse response or by its frequency response. We can implement Eqn. (1.27) in the computer using the DFT, which is only possible for FIR filters.

### 1.3.3 FIR Filters and Symmetry

Filters whose impulse response has a finite support are called finite-impulse response (FIR) filters. A highly desirable property of FIR filters is that they can be designed to have zero or linear phase.

Implementing (1.27) using the DFT yields circular convolution in the space domain, since

$$H(k_1,k_2)S(k_1,k_2) \leftrightarrow h(n_1,n_2) \circledast\circledast s(n_1,n_2)$$

Circular convolution is defined between two periodic signals that share a common period. The result $\tilde{g}(n_1,n_2)$ of circular convolution of two rectangularly periodic signals is also periodic with the same period $(N_1, N_2)$, given by

$$\tilde{g}(n_1,n_2) = \sum_{i_1=0}^{N_1-1}\sum_{i_2=0}^{N_2-1}\tilde{h}(n_1-i_1,n_2-i_2)\tilde{s}(i_1,i_2),$$

$$(n_1,n_2) \in [0,N_1-1] \times [0,N_2-1]$$

The circular convolution produces the same result as that of linear convolution if we set the size of the DFTs to at least $(N_1 + M_1 - 1) \times (N_2 + M_2 - 1)$. Hence, both the FIR filter impulse response array and the image array must be padded by zeros. We can summarize the procedure to implement linear convolution in the DFT domain as:

1. Pad both the FIR filter impulse response $h(n_1, n_2)$ and the image $s(n_1, n_2)$ by zeros to obtain $(N_1 + M_1 - 1) \times (N_2 + M_2 - 1)$ arrays.
2. Compute $(N_1 + M_1 - 1) \times (N_2 + M_2 - 1)$ point DFTs of both $h(n_1,n_2)$ and $s(n_1, n_2)$.
3. Multiply $(N_1 + M_1 - 1) \times (N_2 + M_2 - 1)$ arrays $H(k_1, k_2)$ and $S(k_1, k_2)$.
4. Take $(N_1 + M_1 - 1) \times (N_2 + M_2 - 1)$ point IDFT of the product to find the output.

**Symmetric Filters**

Symmetric filters are desirable since they have zero-phase response or linear-phase response. Furthermore, the number of multiplications in the implementation of (1.20) can be reduced using symmetry. Practical image-processing filters are either two-fold, four-fold, or circularly symmetric.

Two-fold rectangular symmetry is also called non-symmetric half-plane symmetry, which is defined by

$$h(n_1,n_2) = h(-n_1,-n_2) \tag{1.28a}$$

The distinct coefficients are depicted by dots in Figure 1.1(c). The remaining coefficients needed to complete the square support are determined by the symmetry.

A more strict form of symmetry is the four-fold rectangular symmetry given by

$$h(n_1, n_2) = h(-n_1, n_2) = h(n_1, -n_2) \tag{1.28b}$$

The support of the distinct coefficients in this case is a quarter-plane.

Circular symmetry is a natural form of symmetry where coefficients are only a function of distance $n_1^2 + n_2^2$ from the origin. A filter is said to have a circularly symmetric impulse response if $h(n_1, n_2)$ is a function of $n_1^2 + n_2^2$. A filter is said to have a circularly symmetric frequency response if $H\left(e^{j\omega_1}, e^{j\omega_2}\right)$ is a function of $\omega_1^2 + \omega_2^2$ for $\sqrt{\omega_1^2 + \omega_2^2} \leq \pi$ and is constant outside this region within $-\pi \leq \omega_1, \omega_2 \leq \pi$. Circular symmetry of $H\left(e^{j\omega_1}, e^{j\omega_2}\right)$ implies circular symmetry of $h(n_1, n_2)$, but not vice versa.

**Example.** Determine the impulse response of the ideal low-pass filter whose frequency response is circularly symmetric given by

$$H\left(e^{j\omega_1}, e^{j\omega_2}\right) = \begin{cases} 1 & \text{if } \sqrt{\omega_1^2 + \omega_2^2} \leq \omega_c \\ 0 & \sqrt{\omega_1^2 + \omega_2^2} > \omega_c \text{ and } 0 \leq |\omega_1|, \ |\omega_2| < \pi \end{cases}$$

Taking the inverse 2D Fourier transform of $H\left(e^{j\omega_1}, e^{j\omega_2}\right)$ yields

$$h(n_1, n_2) = \frac{\omega_c}{2\pi\sqrt{n_1^2 + n_2^2}} J_1\left(\omega_c \sqrt{n_1^2 + n_2^2}\right)$$

where $J_1(x)$ denotes Bessel function of first kind and first order, which can be expressed as a series

$$J_1(x) = \frac{x}{2} - \frac{x^3}{2^3\ 2!} + \frac{x^5}{2^5\ 2!3!} - \frac{x^7}{2^7\ 3!4!} + \cdots$$

Referring to Section 1.2.2, two-fold symmetry of $h(n_1, n_2)$ is sufficient for $H\left(e^{j\omega_1}, e^{j\omega_2}\right)$ to be real; hence, zero phase. If $h(n_1, n_2)$ is shifted so that it is symmetric with respect to some point other than the origin, then it will have linear phase.

## 1.3.4 IIR Filters and Partial Difference Equations

The convolution summation (1.20) cannot be computationally evaluated if the impulse response has infinite extent. Technically, IIR filters cannot be implemented using the DFT method either, since the DFT cannot be defined for infinite-extent

sequences. However, in practice, if the filter-impulse response decays fast enough, it is possible to approximately implement IIR filters in the DFT domain using a sufficiently large DFT size with some tolerable spatial-domain aliasing.

In general, in two or more dimensions, linear IIR systems are governed by partial difference equations given by

$$g(n_1, n_2) = \sum_{i_1} \sum_{i_2} a_{i_1 i_2}\, g(n_1 - i_1, n_2 - i_2) + \sum_{i_1} \sum_{i_2} b_{i_1 i_2}\, s(n_1 - i_1, n_2 - i_2) \tag{1.29}$$

where the output $g(n_1, n_2)$ can be expressed in terms of past outputs recursively. Therefore, IIR filters are also called recursive filters. There are three important key concepts regarding recursive filters: i) recursive computability, ii) stability, and iii) boundary conditions.

### Recursive Computability

In order to compute $g(n_1, n_2)$ given $s(n_1, n_2)$ from (1.29), we need to define a scanning order to parse samples into a 1D order to label them as past, present, and future. Lexicographic order scans all pixels in a line from left to right and then advances to the leftmost sample of the next line and repeats the same process. Then, all samples within a non-symmetric half-plane (NSHP) support about the current pixel (all previous lines and samples to the left on the current line) are considered "past" samples and (1.29) is recursively computable if the coefficient array $\{a_{i_1 i_2}\}$ has NSHP support meaning that the first summation contains only "past" terms.

### Stability

Eqn. (1.29) will yield meaningful results if the 2D IIR filter is stable. Testing stability of 2D recursive filters is beyond the scope of this book, and the reader is referred to [Wds 06].

### Boundary Conditions

In order to compute the output samples $g(n_1, n_2)$, we need boundary conditions around three borders of the output array as depicted in Figure 1.7. The width of the boundary is related to the order of the filter, i.e., the support of the coefficient array $\{a_{i_1 i_2}\}$.

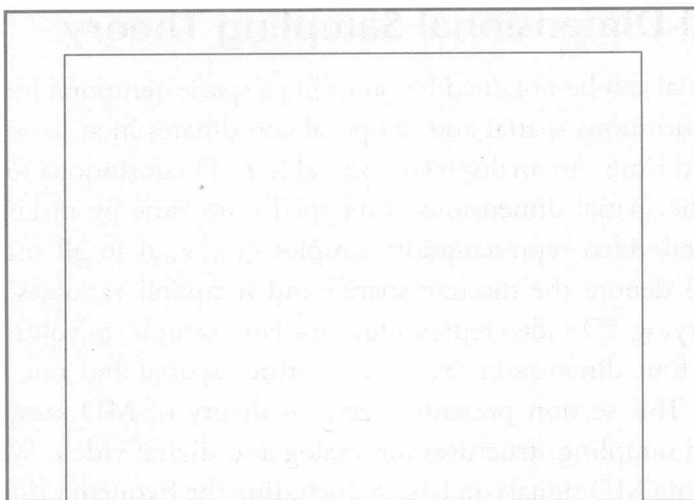


**Figure 1.7** Boundary conditions for 2D recursive filtering.

### Impulse Response

Given a difference equation (1.29), one can compute the impulse response of the filter by setting the input $s(n_1, n_2) = \delta(n_1, n_2)$ and all boundary values equal to zero.

#### Example: 2D Auto-Regressive (AR) Model

A 2D-AR model with NSHP support is a difference equation driven by white noise $w(n_1, n_2)$, typically zero mean Gaussian with variance $\sigma_w^2$, where the coefficient array $\{a_{i_1 i_2}\}$ has NSHP support and all $b_{i_1 i_2} = 0$ except for $b_{00} = 1$. The 2D AR model is given by

$$s(n_1, n_2) = a_{11}\ s(n_1 - 1, n_2 - 1) + a_{01}\ s(n_1, n_2 - 1) + a_{-11}\ s(n_1 + 1, n_2 - 1) + a_{10} s(n_1 - 1, n_2) + w(n_1, n_2)$$

In this case, the width of the boundary is 1 pixel; i.e., the first column, the last column, and the first line (row) of the output $s(n_1, n_2)$ must be known in order to compute the rest of the output samples recursively, given the filter coefficients and the input $w(n_1, n_2)$. As a rule of thumb, the filter is generally stable if

$$\sum_{i_1} \sum_{i_2} a_{i_1 i_2} < 1$$

The 2D-AR model is often used to model the autocorrelation or power spectrum of an image.

## 1.4 Multi-Dimensional Sampling Theory

A 2D-video signal can be obtained by sampling a spatio-temporal intensity function $s_c(x_1, x_2, t)$ of continuous spatial and temporal coordinates in at least one of the spatial variables and time. An analog-video signal is a 1D continuous function of time, where one of the spatial dimensions is mapped onto time by means of a scanning process. A digital-video representation samples $s_c(x_1, x_2, t)$ in all three dimensions, where $(n_1, n_2, k)$ denote the discrete spatial and temporal variables. There are also digital, time-varying 3D-video representations. For example, in volumetric 3D-video representation, four dimensions $(x_1, x_2, x_3, t)$ (three spatial and one temporal) have to be sampled. This section presents a general theory of MD sampling and some commonly used sampling structures for analog and digital video. We introduce the theory of sampling MD signals on lattices including the frequency domain characterization of sampled video signals. We present some examples, including sampling for analog video over 2D lattices and sampling for digital video over 3D and 4D lattices.

In classical signal and image-processing texts [Cro 83, Opp 89], sampling of 1D and 2D signals is often modeled by multiplication of the analog signal with an appropriate impulse train (a periodic sequence of Dirac delta functions), and the frequency domain analysis of sampling is introduced through convolution of the continuous Fourier transform of the analog signal with that of an appropriate impulse train (using the modulation property of the Fourier transform). While this framework can be easily extended to the case of MD sampling on rectangular grids (or other structures where the resulting impulse train is separable), it is not straightforward to study MD sampling on arbitrary periodic structures (e.g., vertically aligned 2:1 interlaced sampling) with this approach. Thus, we adopt the more general lattice framework [Dud 84, Dub 85] to study MD sampling, and present special cases to clarify the concept and notation.

### 1.4.1 Sampling on a Lattice

We begin with the definition of an MD lattice. Let $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_M$ be linearly independent vectors in the MD Euclidean space $\mathrm{R}^{\mathrm{M}}$. A lattice $\Lambda^{\mathrm{M}} \in \mathrm{R}^{\mathrm{M}}$ is the set of all linear combinations of $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_M$ with integer coefficients given by

$$\Lambda^{\mathrm{M}} = \{n_1\mathbf{v}_1 + n_2\mathbf{v}_2 + \ldots + n_M\mathbf{v}_M = \mathbf{V}\mathbf{n} \mid n_1, n_2, \ldots, n_M \in \mathbf{Z}\} \tag{1.30}$$

The set of vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_M$ is called a basis for the lattice $\Lambda^{\mathrm{M}}$, which defines an MD arbitrary periodic sampling structure. An example of a 2D sampling lattice is shown in Figure 1.8.

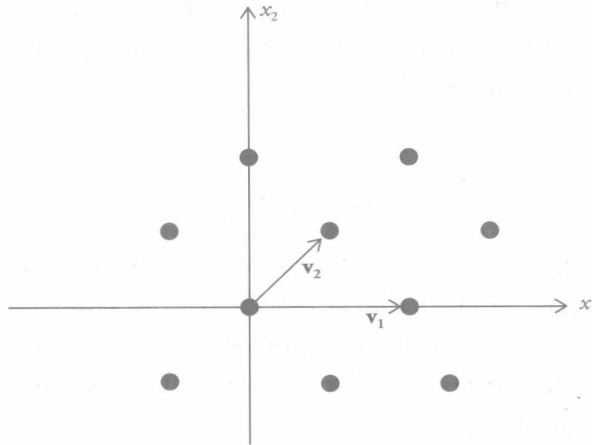



**Figure 1.8** 2D sampling lattice with $M = 2$.

In vector-matrix notation, a lattice $\Lambda^{\mathrm{M}}$ is the set of points defined by

$$\Lambda^{\mathrm{M}} = \{\mathbf{Vn} | \mathbf{n} \in Z^{M}\}$$

where $\mathbf{V}$ is called an $M \times M$ sampling matrix given by

$$\mathbf{V} = [\mathbf{v}_1 \,|\, \mathbf{v}_2 \,|\dots|\, \mathbf{v}_{\mathrm{M}}] \tag{1.31}$$

and $\mathbf{n}^{\mathrm{T}} = [n_1, n_2, \dots, n_{\mathrm{M}}]$. The basis, and thus the sampling matrix, for a given lattice is not unique. In particular, for every sampling matrix $\mathbf{V}$, $\mathbf{EV}$, where $\mathbf{E}$ is an integer matrix with $\det\{\boldsymbol{E}\} = \pm 1$, forms another sampling matrix for $\Lambda^{\mathrm{M}}$. However, the quantity $d(\Lambda^{\mathrm{M}}) = |\det\{\mathbf{V}\}|$ is unique and denotes the reciprocal of the sampling density.

Then, the sampled MD signal can be expressed as

$$\begin{aligned} s(\mathbf{n}) &= s_c(\mathbf{Vn}), \mathbf{n} \in \mathbf{Z}^M \\ &= s_c(\mathbf{x}), \mathbf{x} \in \Lambda^M \end{aligned} \tag{1.32}$$

The most suitable sampling structure for a time-varying image depends on its spatio-temporal frequency content. We present some examples in order to clarify the concept and the notation:

1. *2D rectangular sampling:* The $2 \times 2$ matrix $\mathbf{V}$ is diagonal and applies to both rectangular sampling of still images in the horizontal $x_1$ and vertical $x_2$ directions and progressive (non-interlaced) analog video that has been sampled in

the vertical $x_2$ and temporal $t$ directions. In the former case, the diagonal elements of $\mathbf{V}$ are the horizontal and vertical sampling distances, $\Delta x_1$ and $\Delta x_2$, respectively, and the sample locations are

$$x_1 = n_1 \Delta x_1$$
$$x_2 = n_2 \Delta x_2$$

The sampled signal can be expressed in the unitless coordinates $(n_1, n_2)$ as $s(n_1, n_2) = s_c(n_1 \Delta x_1, n_2 \Delta x_2)$, $(n_1, n_2) \in \mathbf{Z}^2$. In the latter case, an analog-video signal is obtained by sampling the time-varying image intensity distribution in the vertical $x_2$ and temporal $t$ directions by a process known as scanning, and the sample locations are

$$x_2 = n_2 \Delta x_2$$
$$t = k \Delta t$$

Continuous intensity information along each horizontal line is concatenated to form a 1D analog video signal as a function of time. The 2D rectangular sampling grid, which yields progressive analog video, and the associated sampling matrix $\mathbf{V}$ are shown in Figure 1.9, where each dot indicates a continuous line of video perpendicular to the plane of the page.

2. *2D sampling on arbitrary lattices:* It applies to non-rectangular periodic sampling of still images in the $x_1$ and $x_2$ directions, or 2:1 interlaced sampling of analog video in the vertical $x_2$ and temporal $t$ directions. The sampling geometry can be specified by two basis vectors $\mathbf{v}_1 = [v_{11}\ v_{21}]^T$ and $\mathbf{v}_2 = [v_{12}\ v_{22}]^T$ as:

$$x_1 = v_{11} n_1 + v_{12} n_2$$
$$x_2 = v_{21} n_1 + v_{22} n_2$$

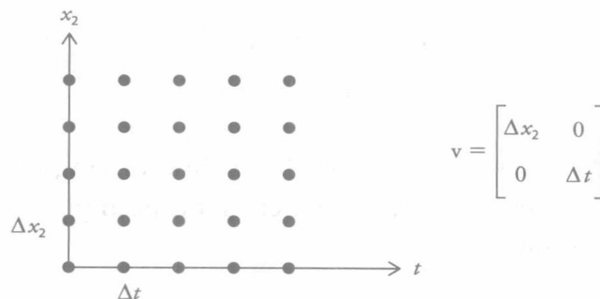



**Figure 1.9** Orthogonal sampling structure for progressive analog video.

In vector-matrix form, we have $\mathbf{x} = \mathbf{V}\mathbf{n}$, where $\mathbf{x} = [x_1\ x_2]^T$, $\mathbf{n} = [n_1\ n_2]^T$, and $\mathbf{V} = [\mathbf{v}_1 \mid \mathbf{v}_2]$ is the sampling matrix. Then, the sampled signal can be expressed as

$$s(\mathbf{n}) = s_c(\mathbf{V}\mathbf{n}), \mathbf{n} \in \mathbf{Z}^2$$

The hexagonal sampling structure, which yields 2:1 interlaced analog video and the associated sampling matrix **V**, is depicted in Figure 1.10.

3. *3D sampling on lattices:* This case applies to sampling of a 3D volume in the $x_1$, $x_2$, and $x_3$ directions, or sampling video signals in the $x_1$, $x_2$, and $t$ directions.

   Digital video can be captured using a digital camera that records samples on an explicit 3D structure or by sampling the analog-video signal in the horizontal direction (along the scan lines), which results in an array of color/intensity samples on an implicit 3D structure. The latter process is known as analog-to-digital conversion.

   Examples of 3D sampling lattices and their corresponding **V** matrices are depicted in Figure 1.11 and Figure 1.12, where each circle indicates a pixel location. The letter "p" inside the pixels indicates "progressive" sampling, where all pixels are sampled at the same time, and letters "o" and "e" indicate odd and even field pixels, respectively, which are sampled with a $\Delta t/2$ sec. time difference. The reader is refered to [Dub 85 ] for further details on MD sampling and sampling structures.

The sampling structures shown here are field or frame instantaneous; i.e., a complete field or frame is acquired at one instant. An alternative strategy is time-sequential sampling, where individual samples are taken one at a time according to

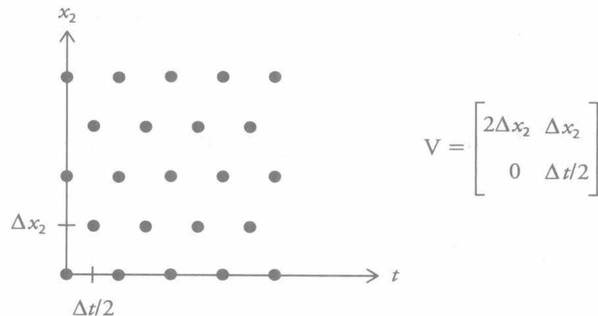



**Figure 1.10** Hexagonal sampling lattice for 2:1 interlaced analog video.

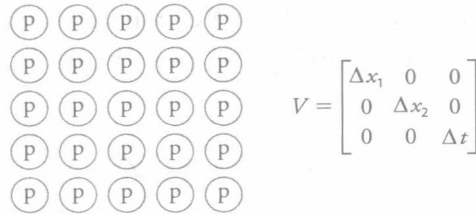



**Figure 1.11** Orthogonal sampling lattice for progressive digital video.

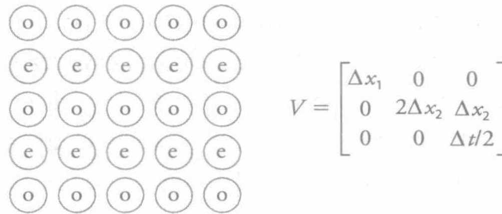



**Figure 1.12** Vertically aligned 2:1 interlace lattice for interlaced digital video.

a prescribed ordering that is repeated after one complete frame. A complete analysis of time-sequential sampling can be found in [Rah 92].

## 1.4.2 Spectrum of Signals Sampled on a Lattice

Let's first recall that the MD continuous Fourier transform (FT) $S_c(F_1, F_2, \ldots, F_M)$ of an analog signal $s_c(x_1, x_2, \ldots, x_M)$ is given by

$$S_c(F_1, F_2, \ldots, F_M) =$$

$$\int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} s_c(x_1, x_2, \ldots, x_M)\, e^{-j2\pi(F_1 x_1 + F_2 x_2 + \ldots + F_M x_M)}\, dx_1\, dx_2 \ldots dx_M$$

where $(F_1, F_2, \ldots, F_M) \in \boldsymbol{R}^{\mathrm{M}}$ and $(x_1, x_2, \ldots, x_M) \in \boldsymbol{R}^{\mathrm{M}}$. The inverse 2D Fourier transform is given by

$$s_c(x_1, x_2, \ldots, x_M) =$$

$$\int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} S_c(F_1, F_2, \ldots, F_M)\, e^{j2\pi(F_1 x_1 + F_2 x_2 + \ldots + F_M x_M)}\, dF_1\, dF_2 \ldots dF_M$$

Here, the spatial-frequency variables $(F_1, F_2, \ldots, F_M)$ have the units cycles/mm and are related to the radian frequencies by a factor of $2\pi$, i.e., $u_i = 2\pi F_i$, $i = 1, \ldots, M$. We can restate the MD Fourier transform relations in compact vector notation as

$$S_c(\mathbf{F}) = \int_{-\infty}^{\infty} s_c(\mathbf{x}) e^{-j2\pi \mathbf{F}^T \mathbf{x}} d\mathbf{x} \tag{1.33}$$

$$s_c(\mathbf{x}) = \int_{-\infty}^{\infty} S_c(\mathbf{F}) e^{j2\pi \mathbf{F}^T \mathbf{x}} d\mathbf{F} \tag{1.34}$$

where $\mathbf{x}^T = [x_1, x_2, \ldots, x_M]$ and $\mathbf{F}^T = [F_1, F_2, \ldots, F_M]$.

The Fourier transform of a discrete signal $s(\mathbf{n})$, sampled on a lattice $\Lambda^M$, is defined, in vector notation, by

$$S(\mathbf{f}) = \sum_{\mathbf{n}=-\infty}^{\infty} s(\mathbf{n}) e^{-j2\pi\, \mathbf{f}^T \mathbf{n}} \tag{1.35}$$

in terms of the unitless (normalized) frequency variables $f_1, f_2, \ldots, f_M$, where $\mathbf{f}^T = [f_1, f_2, \ldots, f_M]$ and $\mathbf{n}^T = [n_1, n_2, \ldots, n_M]$. Note that $\omega_i = 2\pi f_i$, $i = 1, \ldots, M$. The inverse Fourier transform can be expressed as

$$s(\mathbf{n}) = \int_{-1/2}^{1/2} S(\mathbf{f}) e^{j2\pi \mathbf{f}^T \mathbf{n}} d\mathbf{f} \tag{1.36}$$

Recall that the Fourier transform $S(\mathbf{f})$ of a discrete signal is periodic with the fundamental period $f_i < |1/2|$, $\mathrm{i} = 1, \ldots, M$.

In order to quantify the relationship between the Fourier transform $S(\mathbf{f})$ of a signal sampled on a lattice and that of $S_c(\mathbf{F})$ the underlying analog signal, we next define the reciprocal lattice and the unit cell of a lattice. Given a lattice $\Lambda^M$, the set of all vectors $\mathbf{r}$ such that $\mathbf{r}^T\mathbf{x}$ is an integer for all $\mathbf{x} \in \Lambda^M$ is called the reciprocal lattice $\Lambda^{M*}$ of $\Lambda^M$. A basis for $\Lambda^{M*}$ is the set of vectors $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_M$ determined by

$$\mathbf{u}_i^T \mathbf{v}_j = \delta_{ij}, i, j = 1, 2, \ldots, M \tag{1.37}$$

or, equivalently, by

$$\mathbf{U}^T \mathbf{V} = \mathbf{I}_M$$

where $\mathbf{I}_M$ is an $M \times M$ identity matrix. We will see that the Fourier transform of a signal sampled on a lattice $\Lambda^M$ consists of sum of periodic replications of the spectrum of the analog signal on the reciprocal lattice $\Lambda^{M*}$.

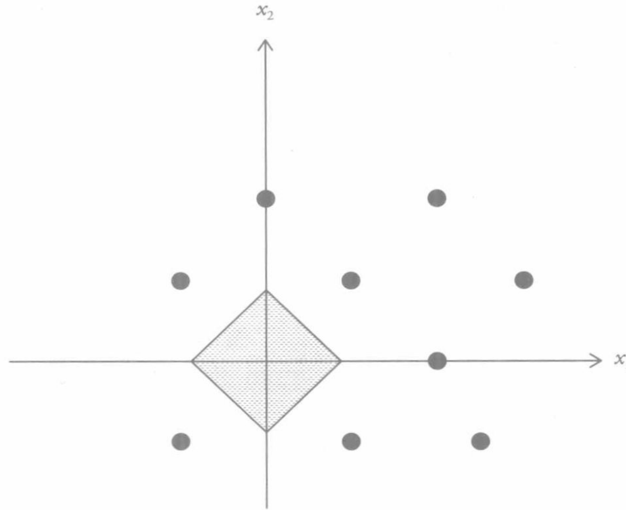



**Figure 1.13** Voronoi cell of a 2D lattice.

The definition of the unit cell of a lattice is not unique. Here, we define the Voronoi cell of a lattice as a unit cell. The Voronoi cell, depicted in Figure 1.13, is the set of all points that is closer to the origin than to any other sample point. It corresponds to the fundamental period of the lattice.

## 1.4.3 Nyquist Criterion for Sampling on a Lattice

In this section, we study frequency domain analysis of sampling on MD lattices. We start by substituting Eqn. (1.34) into (1.32) to obtain

$$s(\mathbf{n}) = s_c(\mathbf{Vn}) = \int_{-\infty}^{\infty} S_c(\mathbf{F}) e^{j\,2\pi\,\mathbf{F}^{\mathrm{T}}\mathbf{Vn}} \mathbf{dF}$$

After the change of variables $\mathbf{f} = \mathbf{V}^T\,\mathbf{F}$, we have

$$s(\mathbf{n}) = \frac{1}{|\det(\mathbf{V})|} \int_{-\infty}^{\infty} S_c(\mathbf{Uf}) e^{j\,2\pi\,\mathbf{f}^{\mathrm{T}}\mathbf{n}} \mathbf{df}$$

where $\mathbf{U} = (\mathbf{V}^{\mathrm{T}})^{-1}$ is the sampling matrix of the reciprocal lattice $\Lambda^{M^*}$ and $\mathbf{df} = |\det(\mathbf{V})|\,\mathbf{dF}$.

Expressing the integration over the entire $\mathbf{f}$ plane as a sum of integrations over the unit-cell squares $(-1/2, 1/2) \times (-1/2, 1/2)$, we obtain

$$s(\mathbf{n}) = \frac{1}{|\det(\mathbf{V})|} \sum_{\mathbf{k}} \int_{-1/2}^{1/2} S_c(\mathbf{U}(\mathbf{f}-\mathbf{k})) e^{j\,2\pi\,\mathbf{f}^T\mathbf{n}} e^{-j2\pi\,\mathbf{k}^T\mathbf{n}}\, \mathbf{df}$$

where $e^{-j\,2\pi\,\mathbf{k}^T\mathbf{n}} = 1$ for $\mathbf{k}$, an integer-valued vector (by definition of the reciprocal lattice). Finally, comparing this expression with (1.36), we conclude that

$$S(\mathbf{f}) = \frac{1}{|\det(\mathbf{V})|} \sum_{\mathbf{k}} S_c(\mathbf{U}(\mathbf{f}-\mathbf{k})) \tag{1.38}$$

where $\mathbf{U}$ is the periodicity matrix in the frequency domain that satisfies $\mathbf{U}^T\mathbf{V} = \mathbf{I}_M$ and $\mathbf{I}_M$ is the $M \times M$ identity matrix. The periodicity matrix can be expressed as $\mathbf{U} = [\mathbf{u}_1 \,|\, \mathbf{u}_2 \,|\ldots|\, \mathbf{u}_M]$, where $\mathbf{u}_1, \ldots, \mathbf{u}_M$ are the basis vectors of the reciprocal lattice.

The MD signal sampled on a lattice can alternatively be expressed in terms of continuous variables as

$$s_p(\mathbf{x}) = s_c(\mathbf{x}) \sum_{\mathbf{n}\in\mathbf{Z}^M} \delta(\mathbf{x}-\mathbf{Vn}) = \sum_{\mathbf{n}\in\mathbf{Z}^M} s(\mathbf{n})\delta(\mathbf{x}-\mathbf{Vn}) \tag{1.39}$$

The Fourier transform $S_p(\mathbf{F})$ of the sampled signal $s_p(\mathbf{x})$ in terms of that of the continuous signal $S_c(\mathbf{F})$ can be obtained from (1.38) by a change of variables as

$$S_p(\mathbf{F}) = \frac{1}{|\det(\mathbf{V})|} \sum_{\mathbf{k}} S_c(\mathbf{F}-\mathbf{Uk}) \tag{1.40}$$

As expected, the Fourier transform of the sampled signal is the sum of an infinite number of replications of the Fourier transform of the continuous signal, shifted according to the reciprocal lattice $\Lambda^{M*}$. This is illustrated in Figure 1.14 for the case of a circularly bandlimited signal.

### Special Case: Two-Dimensional Rectangular Sampling

Rectangular sampling is a special case of lattice sampling where the matrices $\mathbf{V}$ and $\mathbf{U}$ are diagonal. The classical approach of multiplication by an impulse train and use of the modulation property of the Fourier transform would actually suffice in this case (see Exercise 1.7). However, we take this special case to demonstrate the lattice analysis in a step-by-step fashion. We first substitute (1.34) into (1.32) with $M = 2$, and evaluate $x_1$ and $x_2$ at $x_1 = n_1\Delta x_1$ and $x_2 = n_2\Delta x_2$ to obtain

$$s(n_1, n_2) = \iint_{-\infty}^{\infty} S_c(F_1, F_2) e^{j2\pi(F_1 n_1 \Delta x_1 + F_2 n_2 \Delta x_2)}\, dF_1\, dF_2$$

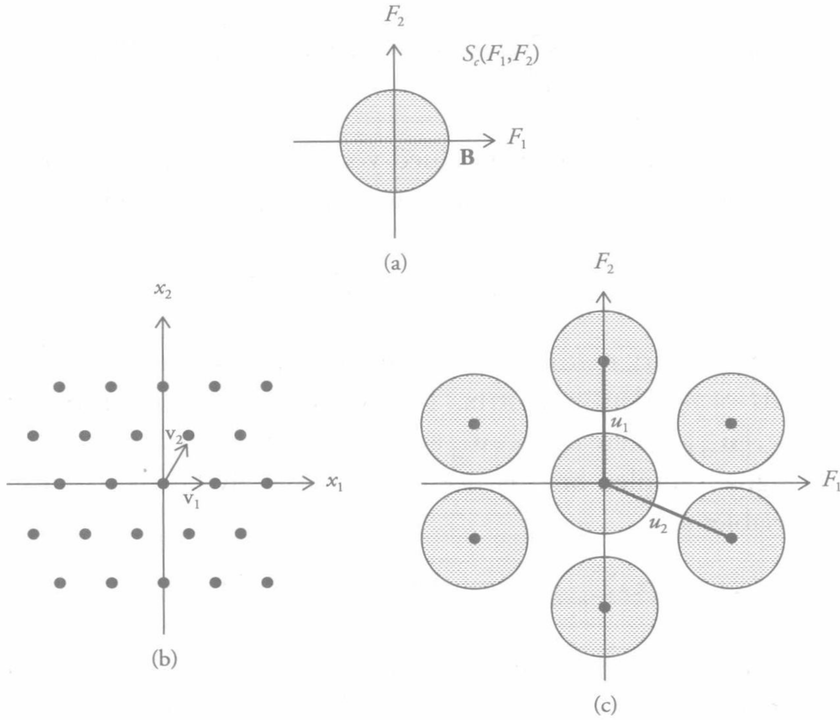



**Figure 1.14** Sampling on an arbitrary 2D periodic grid: (a) spectral support of the continuous image; (b) the sampling grid; (c) spectral support of the sampled image.

After a change of variables $f_1 = F_1 \Delta x_1$ and $f_2 = F_2 \Delta x_2$, we have

$$s(n_1, n_2) = \frac{1}{\Delta x_1 \Delta x_2} \iint_{-\infty}^{\infty} S_c\left(\frac{f_1}{\Delta x_1}, \frac{f_2}{\Delta x_2}\right) e^{j2\pi(f_1 n_1 + f_2 n_2)}\, df_1\, df_2$$

Next, we break the integration over the entire $(f_1, f_2)$ plane into a sum of integrals over unit cells denoted by $SQ(k_1, k_2)$:

$$s(n_1, n_2) = \frac{1}{\Delta x_1 \Delta x_2} \sum_{k_1} \sum_{k_2} \iint_{SQ(k_1,k_2)} S_c\left(\frac{f_1}{\Delta x_1}, \frac{f_2}{\Delta x_2}\right) e^{j2\pi(f_1 n_1 + f_2 n_2)} df_1 df_2$$

where $SQ(k_1, k_2)$ is defined as

$$-\frac{1}{2} + k_1 \le f_1 < \frac{1}{2} + k_1 \text{ and } -\frac{1}{2} + k_2 \le f_2 < \frac{1}{2} + k_2$$

Another change of variables, $f_1 = f_1 - k_1$ and $f_2 = f_2 - k_2$, shifts all unit cells $SQ(k_1, k_2)$ down to the fundamental period $\left(-\tfrac{1}{2}, \tfrac{1}{2}\right) \times \left(-\tfrac{1}{2}, \tfrac{1}{2}\right)$,

$$s(n_1, n_2) =$$

$$\frac{1}{\Delta x_1 \Delta x_2} \sum_{k_1} \sum_{k_2} \iint_{-1/2}^{1/2} S_c\left(\frac{f_1 - k_1}{\Delta x_1}, \frac{f_2 - k_2}{\Delta x_2}\right) e^{j2\pi(f_1 n_1 + f_2 n_2)}\, df_1\, df_2 \tag{1.41}$$

since $e^{-j2\pi(k_1 n_1 + k_2 n_2)} = 1$ for $k_1, k_2, n_1, n_2$ integers. Now, rewriting (1.36) for $M = 2$, we have

$$s(n_1, n_2) = \iint_{-1/2}^{1/2} S(f_1, f_2) e^{j2\pi(f_1 n_1 + f_2 n_2)}\, df_1\, df_2 \tag{1.42}$$

and comparing the expressions (1.41) and (1.42), we conclude that

$$S(f_1, f_2) = \frac{1}{\Delta x_1 \Delta x_2} \sum_{k_1} \sum_{k_2} S_c\left(\frac{f_1 - k_1}{\Delta x_1}, \frac{f_2 - k_2}{\Delta x_2}\right) \tag{1.43}$$

Note that (1.43) is a special case of (1.38) where $M = 2$ and the $2 \times 2$ matrix $\mathbf{V}$ is diagonal. We see that, as a result of sampling, the spectrum of the continuous signal replicates in the 2D frequency plane according to (1.43). The case when the continuous signal is bandlimited with a circular spectral support of radius $B < \max\left\{1/(2\Delta x_1), 1/(2\Delta x_2)\right\}$ is depicted in Figure 1.15.

If the image is not bandlimited or the sampling intervals $\Delta x_1$ and $\Delta x_2$ do not satisfy the conditions of the Nyquist sampling theorem, then the replications overlap with each other, which results in aliasing. This latter case is illustrated in Figure 1.16.

In all sampling problems, it is also possible to define the sampled signal in terms of the continuous coordinate variables by using the 2D Dirac delta signal as

$$\begin{aligned} s_p(x_1, x_2) &= s_c(x_1, x_2) \sum_{n_1} \sum_{n2} \delta\left(x_1 - n_1 \Delta x_1, x_2 - n_2 \Delta x_2\right) \\ &= \sum_{n_1} \sum_{n_2} s_c(n_1 \Delta x_1, n_2 \Delta x_2) \delta(x_1 - n_1 \Delta x_1, x_2 - n_2 \Delta x_2) \\ &= \sum_{n_1} \sum_{n_2} s(n_1, n_2) \delta(x_1 - n_1 \Delta x_1, x_2 - n_2 \Delta x_2) \end{aligned} \tag{1.44}$$

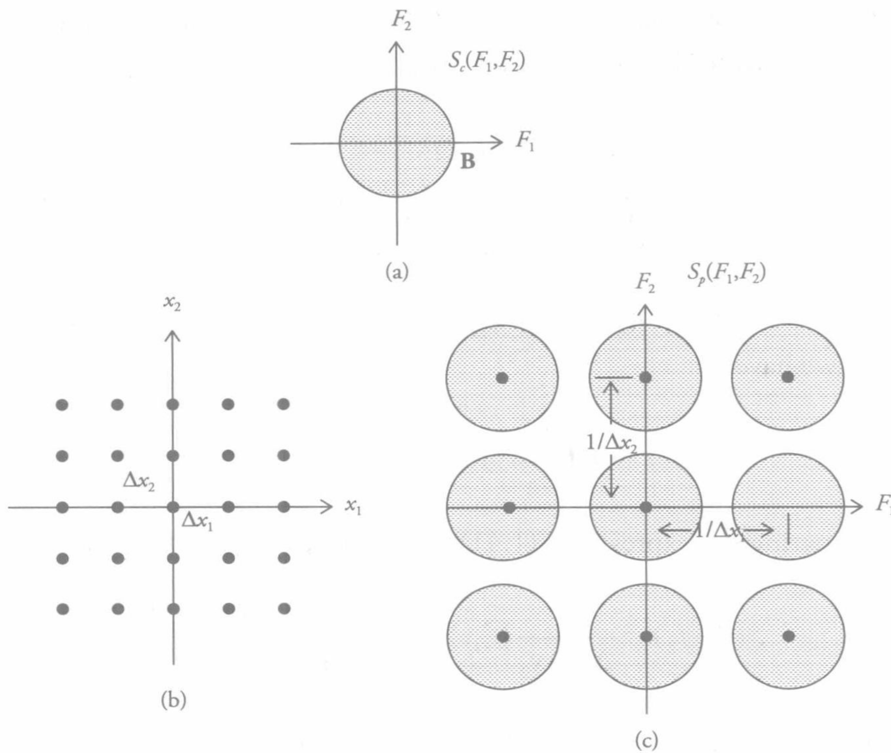



**Figure 1.15** Sampling on a 2D rectangular grid: (a) spectral support of the continuous image; (b) the sampling grid; (c) spectral support of the sampled image.

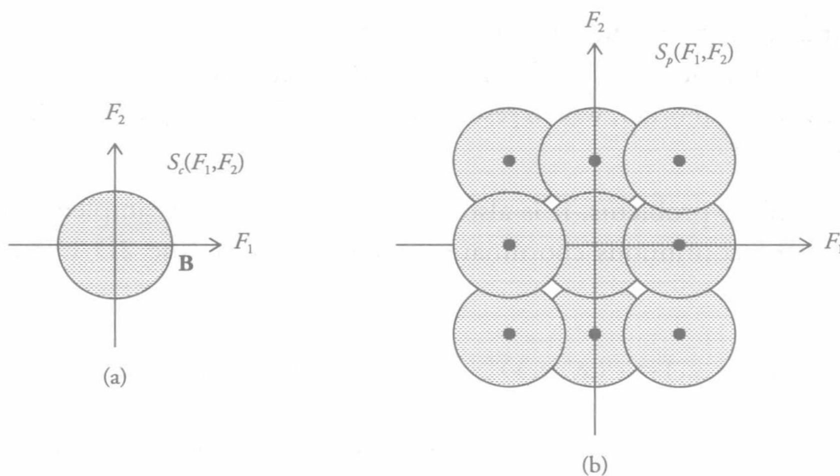



**Figure 1.16** Illustration of aliasing, when Nyquist sampling rate is violated.

Observe that $s_p(x_1, x_2)$ is indeed a sampled signal because of the presence of the 2D Dirac delta function in its definition. Then, the relationship between the Fourier transform $S_p(F_1, F_2)$ of $s_p(x_1, x_2)$ and that of the analog signal can be obtained from Eqn. (1.43), by a change of variables $f_1 = F_1 \Delta x_1$ and $f_2 = F_2 \Delta x_2$, as

$$S_p(F_1, F_2) = S(F_1 \Delta x_1, F_2 \Delta x_2) = \frac{1}{\Delta x_1 \Delta x_2} \sum_{k_1} \sum_{k_2} S_c\left(F_1 - \frac{k_1}{\Delta x_1}, F_2 - \frac{k_2}{\Delta x_2}\right) \quad (1.45)$$

Note that $S_p(F_1, F_2)$ is periodic with the fundamental period $F < \left|\frac{1}{2\Delta x_i}\right|$, $i = 1, 2$.

## 1.4.4 Reconstruction from Samples on a Lattice

Various digital-video systems have different spatio-temporal resolution requirements, which necessitate sampling structure conversion. The sampling structure conversion problem, which is treated in Section 1.5, can alternatively be posed as reconstruction of the underlying continuous spatio-temporal video, followed by its resampling on the desired spatio-temporal lattice. Thus, we briefly discuss reconstruction of a continuous video signal from its samples.

The reconstructed time-varying image $s_r(\mathbf{x}, t)$ can be obtained through the ideal low-pass filtering operation

$$S_r(\mathbf{F}) = \begin{cases} \det(\mathbf{V}) S(\mathbf{V}^T \mathbf{F}) & \text{for } \mathbf{F} \in \boldsymbol{P} \\ 0 & \text{otherwise} \end{cases} \quad (1.46)$$

Here, the passband of the ideal low-pass filter is determined by the unit cell $\boldsymbol{P}$ of the reciprocal sampling lattice, which is depicted by the dotted lines in Figure 1.17.

Taking the inverse Fourier transform, we have the reconstructed time-varying image:

$$s_r(\mathbf{x}, t) = \sum_{(\mathbf{n},k) \in \mathbf{Z}^3} s(\mathbf{n}, k) h\left(\begin{bmatrix} \mathbf{x} \\ t \end{bmatrix} - \mathbf{V}\begin{bmatrix} \mathbf{n} \\ k \end{bmatrix}\right)$$

$$= \sum_{(\mathbf{z},\tau) \in \Lambda^3} s_p(\mathbf{z}, \tau) h\left(\begin{bmatrix} \mathbf{x} \\ t \end{bmatrix} - \begin{bmatrix} \mathbf{z} \\ \tau \end{bmatrix}\right) \quad (1.47)$$

where

$$h(\mathbf{x}, t) = |\det(\mathbf{V})| \int_{\boldsymbol{P}} e^{j2\pi \mathbf{F}^T \begin{bmatrix} \mathbf{x} \\ t \end{bmatrix}} d\mathbf{F} \quad (1.48)$$

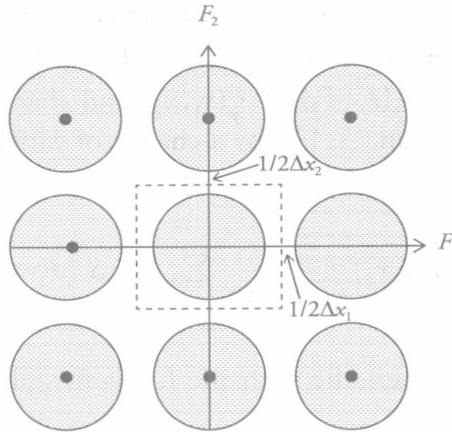



**Figure 1.17** Passband of the ideal reconstruction filter.

is the impulse response of the ideal bandlimited spatio-temporal interpolation filter for the sampling structure used. Unlike the case of rectangular sampling, this integral, in general, cannot be reduced to a simple closed-form expression. As expected, exact reconstruction of a continuous signal from its samples on a lattice $\mathbf{\Lambda}^3$ is possible if the signal spectrum is confined to a unit cell $\boldsymbol{P}$ of the reciprocal lattice.

## 1.5 Sampling Structure Conversion

Various digital-video systems, ranging from ultra high-definition TV to mobile video, have different spatio-temporal resolution requirements leading to the emergence of different format standards. The task of converting digital video from one format to another is referred to as standards conversion. Effective standards conversion enables exchange of information among various digital-video systems, employing different format standards, to ensure their interoperability.

Standards conversion is a sampling structure conversion problem, i.e., a spatio-temporal interpolation/decimation problem. In theory, sampling structure conversion can be treated in two steps: reconstruction of the underlying continuous spatio-temporal signal, followed by its resampling on the desired MD sampling structure. Here, we introduce an all-digital formulation, where general sampling structure conversion from an MD input lattice to an MD output lattice is posed as an MD digital signal-processing problem with or without taking advantage of the temporal redundancy in the video. This section only introduces the general

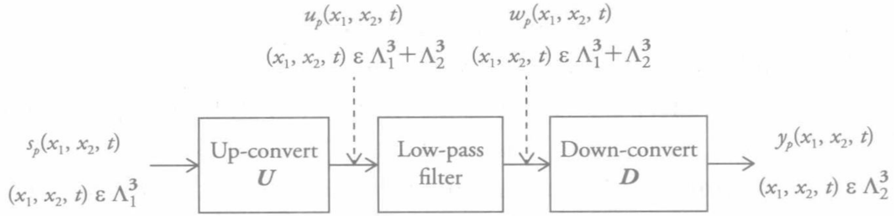



**Figure 1.18** Decomposition of the system for sampling structure conversion.

problem formulation without going into the specifics of the filters involved. Some practical image decimation and interpolation filters are discussed in Chapter 3. Video-filtering methods, including intra-frame/field and inter-frame/field methods, which implicitly or explicitly use interframe motion information, are presented in Chapter 6.

In order to define the general sampling structure conversion problem, we need to define the sum of two lattices, shown in Figure 1.18, as well as the intersection of two lattices. We define the sum of two lattices as

$$\Lambda_1^M + \Lambda_2^M = \left\{\mathbf{x}_1 + \mathbf{x}_2 \mid \mathbf{x}_1 \in \Lambda_1^M \text{ and } \mathbf{x}_2 \in \Lambda_2^M\right\} \tag{1.49}$$

Thus, the sum of two lattices can be found by adding each point in one lattice to every point of the other lattice. We can also define the intersection of two lattices as

$$\Lambda_1^M \cap \Lambda_2^M = \left\{\mathbf{x} \mid \mathbf{x} \in \Lambda_1^M \text{ and } \mathbf{x} \in \Lambda_2^M\right\} \tag{1.50}$$

The intersection $\Lambda_1^M \cap \Lambda_2^M$ is the largest lattice that is a sublattice of both $\Lambda_1^M$ and $\Lambda_2^M$, and the sum $\Lambda_1^M + \Lambda_2^M$ is the smallest lattice that contains both $\Lambda_1^M$ and $\Lambda_2^M$ as sublattices.

The up-conversion from $\Lambda_1^M$ to $\Lambda_1^M + \Lambda_2^M$ is defined as follows:

$$u_p(\mathbf{x}) = \boldsymbol{U} s_p(\mathbf{x}) = \begin{cases} s_p(\mathbf{x}) & \mathbf{x} \in \Lambda_1^M \\ \mathbf{0} & \mathbf{x} \notin \Lambda_1^M \text{ and } \mathbf{x} \in \Lambda_1^M + \Lambda_2^M \end{cases} \tag{1.51}$$

and down-conversion from $\Lambda_1^M + \Lambda_2^M$ to $\Lambda_2^M$ as

$$y_p(\mathbf{x}) = \mathbf{D} w_p(\mathbf{x}) = w_p(\mathbf{x}) \ \mathbf{x} \in \Lambda_2^M \tag{1.52}$$

The low-pass filter is applied on the lattice $\Lambda_1^M + \Lambda_2^M$, which has a higher sampling density than both $\Lambda_1^M$ and $\Lambda_2^M$. By definition, this filter will be shift-invariant if the output gets shifted by a vector $\mathbf{p}$ when the input is shifted by $\mathbf{p}$. Thus, we need $\mathbf{p} \in \Lambda_1^M \cap \Lambda_2^M$. This condition is satisfied if $\Lambda_1^M \cap \Lambda_2^M$ is a lattice; i.e., $\mathbf{V}_1^{-1}\mathbf{V}_2$ is a matrix of rational numbers. This requirement is the counterpart of $L/M$ needing to be a rational number in the case of 1D sampling rate change problems.

The linear shift-invariant filtering operation on $\Lambda_1^M + \Lambda_2^M$ can be expressed as

$$w_p(\mathbf{x}) = \sum_{\mathbf{z} \in \Lambda_1^M + \Lambda_2^M} u_p(\mathbf{z}) h_p(\mathbf{x} - \mathbf{z}) \quad \mathbf{x} \in \Lambda_1^M + \Lambda_2^M \tag{1.53}$$

However, by the definition of the up-sampling operator, $u_p(\mathbf{x}) = s_p(\mathbf{x})$ for $\mathbf{x} \in \Lambda_1^M$ and zero otherwise; thus,

$$w_p(\mathbf{x}) = \sum_{\mathbf{z} \in \Lambda_1^M} s_p(\mathbf{z})\, h_p(\mathbf{x} - \mathbf{z}) \quad \mathbf{x} \in \Lambda_1^M + \Lambda_2^M$$

After the down-sampling operation,

$$y_p(\mathbf{x}) = \sum_{\mathbf{z} \in \Lambda_1^M} s_p(\mathbf{z}) h_p(\mathbf{x} - \mathbf{z}) \quad \mathbf{x} \in \Lambda_2^M \tag{1.54}$$

The frequency response of the filter is periodic with the main period determined by the unit cell of $\left(\Lambda_1^M + \Lambda_2^M\right)^*$. In order to avoid aliasing, the passband of the interpolation/anti-alias (low-pass) filter is restricted to the smaller of the Voronoi cells of $\left(\Lambda_1^M\right)^*$ and $\left(\Lambda_2^M\right)^*$. Sampling lattice conversion is illustrated by the following examples.

### Example: Conversion from $\Lambda_1^2$ to $\Lambda_2^2$ [Dub 85]

We consider a 2D sampling lattice conversion from the input lattice $\Lambda_1^2$ to the output lattice $\Lambda_2^2$, which is shown in Figure 1.19 along with the sampling matrices, $\mathbf{V}_1$ and $\mathbf{V}_2$. The sampling densities of $\Lambda_1^2$ and $\Lambda_2^2$ are inversely proportional to the determinants of $\mathbf{V}_1$ and $\mathbf{V}_2$, given by

$$\det(\mathbf{V}_1) = 2\Delta x_1\, \Delta x_2 \quad \text{and} \quad \det(\mathbf{V}_2) = 4\Delta x_1\, \Delta x_2$$

Hence, the sampling density of the output lattice $\Lambda_2^2$ is one half of that of the input lattice $\Lambda_1^2$. Also shown in Figure 1.19 are the lattices $\Lambda_1^2 + \Lambda_2^2$ and

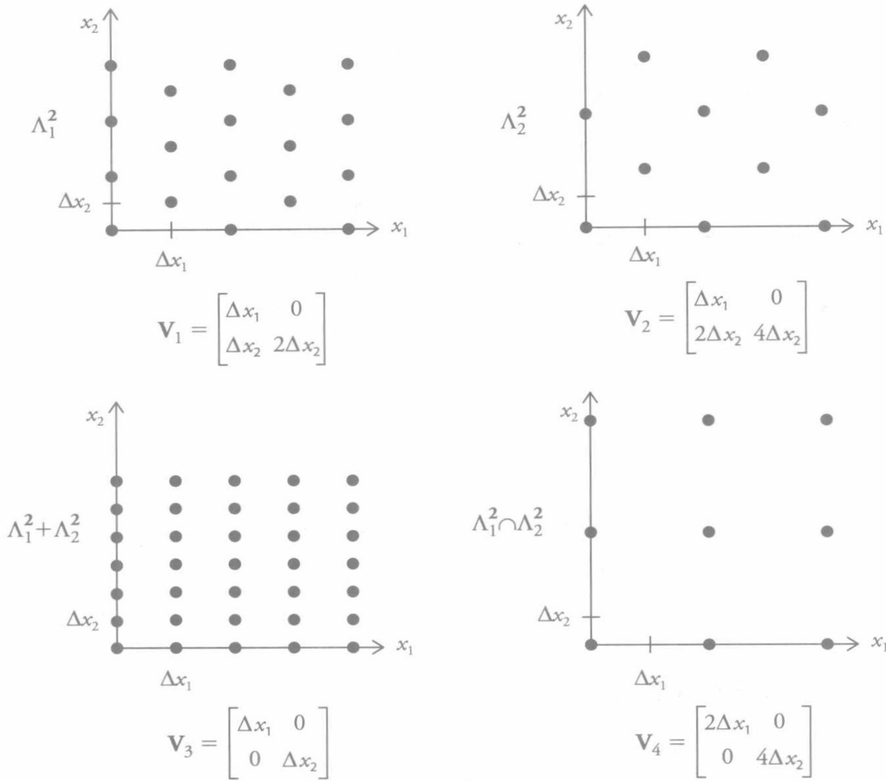



**Figure 1.19** Lattices $\Lambda_1^2, \Lambda_2^2, \Lambda_1^2+\Lambda_2^2$, and $\Lambda_1^2 \cap \Lambda_2^2$ [Dub 85]. (© 1985 IEEE)

$\Lambda_1^2 \cap \Lambda_2^2$, along with their sampling matrices, $\mathbf{V}_3$ and $\mathbf{V}_4$, which can be used to define the sampling conversion factor

$$Q = \left(\Lambda_1^2 + \Lambda_2^2 : \Lambda_1^2\right) = \left(\Lambda_2^2 : \Lambda_1^2 \cap \Lambda_2^2\right) = 2$$

Note that $\Lambda_1^2 + \Lambda_2^2$ is obtained by adding all $\mathbf{x}_1 \in \Lambda_1^2$ to $\mathbf{x}_2 \in \Lambda_2^2$, and $\Lambda_1^2 \cap \Lambda_2^2$ contains all points that are elements of both lattices.

Because we have a down-conversion problem, by a factor of 2, anti-alias filtering must be performed on the lattice $\Lambda_1^2 + \Lambda_2^2$. The fundamental period of the filter-frequency response is given by the unit cell of $\left(\Lambda_1^2 + \Lambda_2^2\right)^*$, which is indicated by the dotted lines in Figure 1.20. Note that the unit cell of $\left(\Lambda_1^2 + \Lambda_2^2\right)^*$ is determined by the matrix $\mathbf{U}_3 = \left(\mathbf{V}_3^T\right)^{-1}$. In order to avoid aliasing, the passband of the low-pass filter performed on the lattice $\Lambda_1^2 + \Lambda_2^2$ must be restricted to the Voronoi cell of $\left(\Lambda_2^2\right)^*$, which is determined by the matrix $\mathbf{U}_2 = \left(\mathbf{V}_2^T\right)^{-1}$ and has a hexagonal shape.

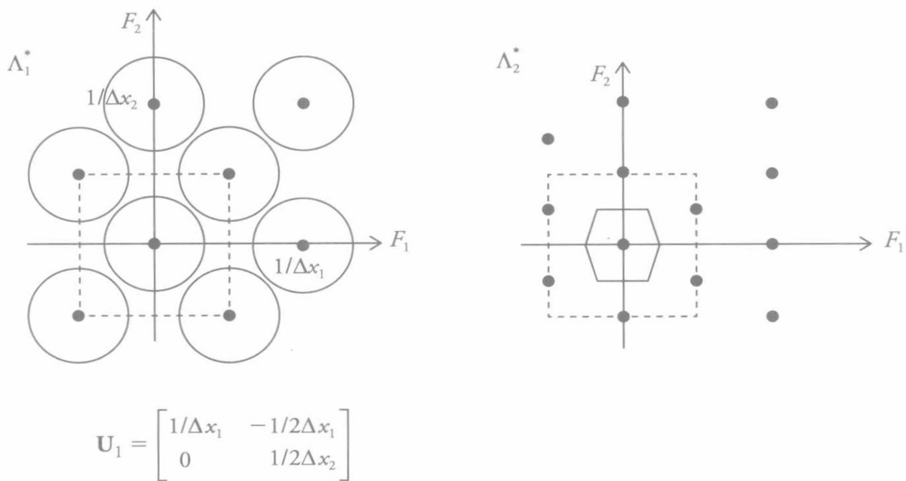



**Figure 1.20** Spectral support of $s(\boldsymbol{x})$ with the periodicity matrix $\boldsymbol{U}_1$, and the support of the frequency response of the filter [Dub 85]. (© 1985 IEEE)

## Example: De-interlacing

De-interlacing refers to conversion from an interlaced sampling grid (input) to a progressive grid (output), as shown in Figure 1.21(a) and (b), respectively. The sampling matrices for the input and output grids are

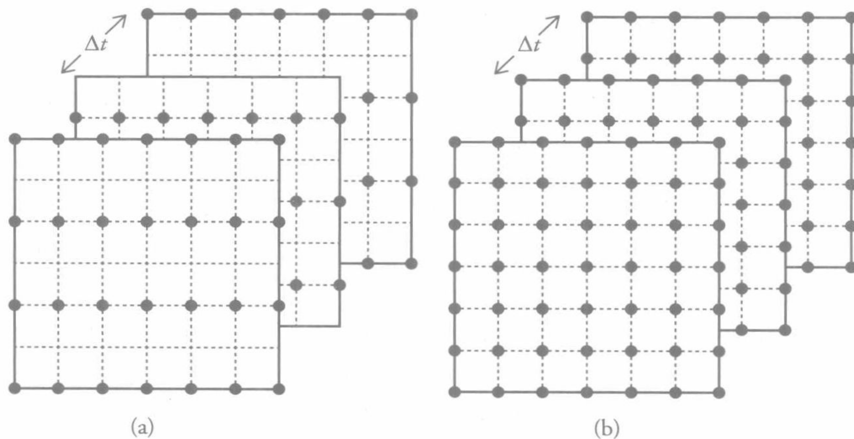



**Figure 1.21** Interlaced to progressive conversion: (a) input; (b) output lattices.

$$\mathbf{V}_{in} = \begin{bmatrix} \Delta x_1 & 0 & 0 \\ 0 & 2\Delta x_2 & \Delta x_2 \\ 0 & 0 & \Delta t \end{bmatrix}$$

and

$$\mathbf{V}_{out} = \begin{bmatrix} \Delta x_1 & 0 & 0 \\ 0 & \Delta x_2 & 0 \\ 0 & 0 & \Delta t \end{bmatrix}$$

respectively. Note that $|\det \mathbf{V}_{\text{in}}| = 2|\det \mathbf{V}_{\text{out}}|$; hence, we have a spatio-temporal interpolation problem by a factor of 2. The interpolation can be achieved by zero filling followed by ideal low-pass filtering. The passband of the ideal low-pass filter should be restricted to the unit cell of the reciprocal lattice of the output sampling lattice, which is given by

$$\left(-\frac{1}{2\Delta x_1}, \frac{1}{2\Delta x_1}\right) \times \left(-\frac{1}{2\Delta x_2}, \frac{1}{2\Delta x_2}\right) \times \left(-\frac{1}{2\Delta t}, \frac{1}{2\Delta t}\right)$$

In spatio-temporal sampling structure down-conversion without motion compensation, there is a tradeoff between allowed aliasing errors and loss of resolution (blurring) due to anti-alias (low-pass) filtering prior to down-conversion. When anti-alias filtering has been used prior to down-conversion, the resolution that is lost cannot be recovered by subsequent interpolation. We present motion-compensated filtering methods to incorporate interframe motion information to sampling structure conversion in Chapter 6. Motion-compensated interpolation makes it possible to recover full-resolution frames by up-conversion of previously down-converted frames if no anti-alias filtering has been applied in the down-conversion process.

## References

[Cro 83] R. E. Crochiere and L. R. Rabiner, *Multirate Digital Signal Processing*, Englewood Cliffs, NJ: Prentice Hall, 1983.

[Dub 85] E. Dubois, "The sampling and reconstruction of time-varying imagery with application in video systems," Proc. of the IEEE, vol. 73, no. 4, pp. 502–522, Apr. 1985.

[Dud 84] D. E. Dudgeon and R. M. Mersereau, *Multidimensional Digital Signal Processing*, Englewood Cliffs, NJ: Prentice Hall, 1984.

[Lim 90] J. S. Lim, *Two-Dimensional Signal and Image Processing*, Englewood Cliffs, NJ: Prentice Hall, 1990.

[Opp 89] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, Englewood Cliffs, NJ: Prentice Hall, 1989.

[Rah 92] M. A. Rahgozar and J. P. Allebach, "A general theory of time-sequential sampling," Signal Proc., vol. 28, no. 3, pp. 253–270, Sep. 1992.

[Wds 06] J. W. Woods, *Multidimensional Signal, Image & Video Processing and Coding*, Salt Lake City, UT: Academic Press, 2006.

# Exercises

## Problem Set 1

**1.1** The signal

$$s(n_1,n_2)=\delta_1(n_1-n_2)=\begin{cases}1 & \text{if } n_1=n_2\\ 0 & \text{otherwise}\end{cases}$$

where $\delta_1(n)$ is a 1D impulse, called a line impulse. Draw this signal and find the angle it makes with the horizontal axis. Can you write other line impulse signals with other angular orientations without a gap in the line?

**1.2** Draw the signal

$$s(n_1,n_2) = u_{\mathrm{HP}}(n_1,n_2)\, u_1(n_1 - n_2)$$

where $u_1(n)$ is a 1D unit step. Is this signal separable? Why or why not?

**1.3** Let a periodic signal satisfy

$$s(n_1,n_2) = s(n_1+3, n_2+5) = s(n_1-2, n_2-3)$$

Find a periodicity matrix **N**. Is it rectangularly periodic? Why or why not?

**1.4** Find a periodicity matrix for

$$s_1(n_1,n_2)=\sin\left(\frac{2\pi n_1}{8}+\frac{2\pi n_2}{16}\right)$$

and

$$s_2(n_1,n_2) = \sin\left(\frac{2\pi n_1}{8}\right)\sin\left(\frac{2\pi n_2}{16}\right)$$

Is $s_1(n_1,n_2) = s_2(n_1,n_2)$? Explain.

**1.5** Convolve

$$s(n_1,n_2) = \begin{cases} 1 & 0 \le n_1 \le N_1 - 1,\ 0 \le n_2 \le N_2 - 1 \\ 0 & \text{otherwise} \end{cases}$$

with

$$h(n_1,n_2) = \begin{cases} 1/9 & -1 \le n_1 \le 1,\ -1 \le n_2 \le 1 \\ 0 & \text{otherwise} \end{cases}$$

Find the frequency response of this filter.

**1.6** Suppose we wish to convolve an $N_1 \times N_2$ image with a $K_1 \times K_2$ impulse response. We have the option of implementing this as a spatial-domain convolution summation or by multiplication in the Fourier domain. For what values of $N = N_1 + N_2$ and $K = K_1 + K_2$ will the spatial-domain convolution be faster than going through the Fourier domain? Explain.

**1.7** In the case of rectangular sampling, the classical approach of multiplication of the analog signal $s_c(x_1,x_2)$ by a 2D rectangular impulse train,

$$s_p(x_1,x_2) = s_c(x_1,x_2)\cdot \sum_{n_1=-\infty}^{\infty}\sum_{n_2=-\infty}^{\infty} \delta(x_1 - n_1\Delta x_1, x_2 - n_2\Delta x_2)$$

and then use of the modulation property of the 2D continuous Fourier transform would suffice to derive (1.43). Derive (1.43) using the approach outlined above.

**1.8** The expression (1.43) assumes impulse sampling; i.e., the sampling aperture has no physical size. A practical camera has a finite pixel aperture modeled by the impulse response $h_a(x_1,x_2)$. How would you incorporate the effect of the finite aperture size into (1.43)?

**1.9** Suppose a camera samples with 20-micron intervals in both the horizontal and vertical directions. What is the highest spatial frequency in the sampled

image that can be represented with less than 3 dB attenuation if a) $h_a(x_1, x_2)$ is a 2D Dirac delta function and b) $h_a(x_1, x_2)$ is a uniform circle with diameter 20 microns?

**1.10** Strictly speaking, images with sharp spatial edges are not bandlimited. Discuss how you would digitize an image that is not bandlimited.

**1.11** Evaluate the impulse response (1.48) if $\boldsymbol{P}$ is the unit sphere.

**1.12** Find the locations of the spectral replications for the 3D sampling lattices depicted in Figure 1.11 and Figure 1.12.

**1.13** Suppose an image is sampled on a sampling lattice defined by

$$\mathbf{V}_1 = \begin{bmatrix} \Delta x_1 & \Delta x_1/2 \\ 0 & \Delta x_2 \end{bmatrix}.$$

**a.** Compute and show where the replications occur in the spatial-frequency domain.

**b.** Show the support of the frequency response of the ideal conversion filter if we wish to convert this sampling structure to a lattice defined by

$$\mathbf{V}_2 = \begin{bmatrix} \Delta x_1/2 & 0 \\ 0 & 2\Delta x_2 \end{bmatrix}.$$

## MATLAB Exercises

**1.1** *Fourier Magnitude and Phase Relations:* Take two gray-level images, $x[n_1, n_2]$ and $y[n_1, n_2]$. Compute the magnitude and phase functions $|X[k_1, k_2]|$, $P_X[k_1, k_2]$, $|Y[k_1, k_2]|$, and $P_Y[k_1, k_2]$ of their 2D-DFT, respectively, where $X[k_1, k_2] = |X[k_1, k_2]| e^{jP_X[k_1, k_2]}$ and $Y[k_1, k_2] = |Y[k_1, k_2]| e^{jP_Y[k_1, k_2]}$.

**a.** Define two new Fourier transforms $W[k_1, k_2] = 1e^{jP_X[k_1, k_2]}$ and $Z[k_1, k_2] = |X[k_1, k_2]| e^{j0}$. Compute and display the images $w[n_1, n_2]$ and $z[n_1, n_2]$. Print both images. Which one looks more similar to $x[n_1, n_2]$?

**b.** Now define $A[k_1, k_2] = |Y[k_1, k_2]| e^{jP_X[k_1, k_2]}$ and $B[k_1, k_2] = |X[k_1, k_2]| e^{jP_Y[k_1, k_2]}$. Compute and display the images $a[n_1, n_2]$ and $b[n_1, n_2]$. Print both images. What do they look like? Explain what you see.

**1.2** *Frequency Response of Filters, DFT:* Given a 1D filter box filter with length nine samples, specified by h = (1/9) ones (9,1)

**a.** Generate a separable 2D filter impulse response from the above 1D filter. Plot the frequency response of the 2D filter. What kind of filter is it?

**b.** Generate a circularly symmetric 2D filter impulse response. Plot the frequency response of the 2D filter.

**1.3** *Spatial-Frequency Patterns:* Generate the horizontal spatial-frequency pattern

$$s(n_1,n_2) = 127\cos(2\pi k_1 n_1/512) + 128,\ 0 \le n_1 \le 511,\ 0 \le n_2 \le 511$$

as an image. Display this image for different values of $k_1$.

CHAPTER 2

# Digital Images and Video

*Advances in ultra-high-definition and 3D-video technologies as well as high-speed Internet and mobile computing have led to the introduction of new video services.*

Digital images and video refer to 2D or 3D still and moving (time-varying) visual information, respectively. A still image is a 2D/3D spatial distribution of intensity that is constant with respect to time. A video is a 3D/4D spatio-temporal intensity pattern, i.e., a spatial-intensity pattern that varies with time. Another term commonly used for video is image sequence, since a video is represented by a time sequence of still images (pictures). The spatio-temporal intensity pattern of this time sequence of images is ordered into a 1D analog or digital video signal as a function of time only according to a progressive or interlaced scanning convention.

We begin with a short introduction to human visual perception and color models in Section 2.1. Next, we present 2D digital video repre-sentations and a brief summary of current standards in Section 2.2. We introduce 3D digital video display, representations, and standards in Section 2.3. Section 2.4 provides an overview of popular digital video applications, including digital TV, digital cinema, and video streaming. Finally, Section 2.5 discusses factors afecting video quality and quantitative and subjective video-quality assessment.

## 2.1 Human Visual System and Color

Video is mainly consumed by the human eye. Hence, many imaging system design choices and parameters, including spatial and temporal resolution as well as color representation, have been inspired by or selected to imitate the properties of human vision. Furthermore, digital image/video-processing operations, including filtering and compression, are generally designed and optimized according to the specifications of the human eye. In most cases, details that cannot be perceived by the human eye are regarded as irrelevant and referred to as perceptual redundancy.

### 2.1.1 Color Vision and Models

The human eye is sensitive to the range of wavelengths between 380 nm (blue end of the visible spectrum) and 780 nm (red end of the visible spectrum). The cornea, iris, and lens comprise an optical system that forms images on the retinal surface. There are about 100-120 million rods and 7-8 million cones in the retina [Wan 95, Fer 01]. They are receptor nerve cells that emit electrical signals when light hits them. The region of the retina with the highest density of photoreceptors is called the *fovea*. Rods are sensitive to low-light (scotopic) levels but only sense the intensity of the light; they enable night vision. Cones enable color perception and are best in bright (photopic) light. They have bandpass spectral response. There are three types of cones that are more sensitive to short (S), medium (M), and long (L) wavelengths, respectively. The spectral response of S-cones peak at 420 nm, M-cones at 534 nm, and L-cones at 564 nm, with significant overlap in their spectral response ranges and varying degrees of sensitivity at these range of wavelengths specified by the function $m_k(\lambda)$, $k =$ r, g, b, as depicted in Figure 2.1(a).

The perceived color of light $f(x_1, x_2, \lambda)$ at spatial location $(x_1, x_2)$ depends on the distribution of energy in the wavelength $\lambda$ dimension. Hence, color sensation can be achieved by sampling $\lambda$ into three levels to emulate color sensation of each type of cones as:

$$f_k(x_1, x_2) = \int f(x_1, x_2, \lambda) m_k(\lambda) d\lambda \quad k = r, g, b \tag{2.1}$$

where $m_k(\lambda)$ is the wavelength sensitivity function (also known as the color-matching function) of the $k$th cone type or color sensor. This implies that perceived color at any location $(x_1, x_2)$ depends only on three values $f_r$, $f_g$, and $f_b$, which are called the tristimulus values.

It is also known that the human eye has a secondary processing stage whereby the R, G, and B values sensed by the cones are converted into a luminance and two

color-difference (chrominance) values [Fer 01]. The luminance Y is related to the perceived brightness of the light and is given by

$$Y(x_1, x_2) = \int f(x_1, x_2, \lambda)\, l(\lambda)\, d\lambda \tag{2.2}$$

where $l(\lambda)$ is the International Commission on Illumination (CIE) luminous efficiency function, depicted in Figure 2.1(b), which shows the contribution of energy at each wavelength to a standard human observer's perception of brightness. Two chrominance values describe the perceived color of the light. Color representations for color image processing are further discussed in Section 2.2.3.

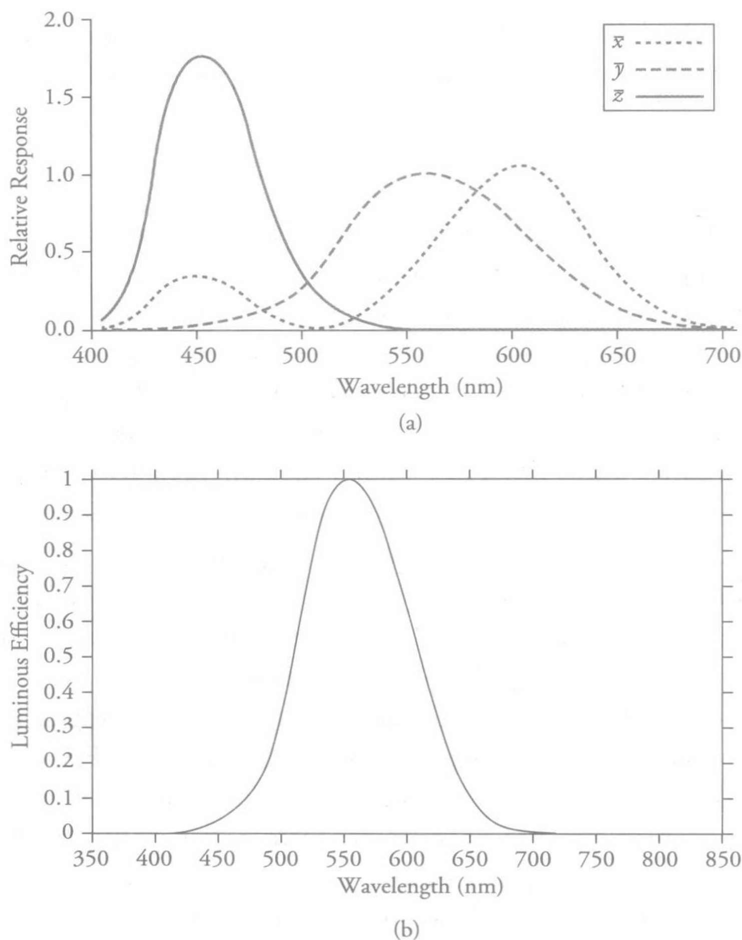



**Figure 2.1** Spectral sensitivity: (a) CIE 1931 color-matching functions for a standard observer with a 2-degree field of view, where the curves $\bar{x}$, $\bar{y}$, and $\bar{z}$ may represent $m_r(\lambda)$, $m_g(\lambda)$, and $m_b(\lambda)$, respectively, and (b) the CIE luminous efficiency function $l(\lambda)$ as a function of wavelength $\lambda$.

Now that we have established that the human eye perceives color in terms of three component values, the next question is whether all colors can be reproduced by mixing three primary colors. The answer to this question is yes in the sense that most colors can be realized by mixing three properly chosen primary colors. Hence, inspired by human color perception, digital representation of color is based on the tri-stimulus theory, which states that all colors can be approximated by mixing three additive primaries, which are described by their color-matching functions. As a result, colors are represented by triplets of numbers, which describe the weights used in mixing the three primaries. All colors that can be reproduced by a combination of three primary colors define the color gamut of a specific device. There are different choices for selecting primaries based on additive and subtractive color models. We discuss the additive RGB and subtractive CMYK color spaces and color management in the following. However, an in-depth discussion of color science is beyond the scope of this book, and interested readers are referred to [Tru 93, Sha 98, Dub 10].

**RGB and CMYK Color Spaces**

The RGB model, inspired by human vision, is an additive color model in which red, green, and blue light are added together to reproduce a variety of colors. The RGB model applies to devices that capture and emit color light such as digital cameras, video projectors, LCD/LED TV and computer monitors, and mobile phone displays. Alternatively, devices that produce materials that reflect light, such as color printers, are governed by the subtractive CMYK (Cyan, Magenta, Yellow, Black) color model. Additive and subtractive color spaces are depicted in Figure 2.2. RGB and CMYK are *device-dependent* color models: i.e., different devices detect or reproduce a given RGB value differently, since the response of color elements (such as filters or dyes) to individual R, G, and B levels may vary among different manufacturers. Therefore, the RGB color model itself does not define absolute *red*, *green*, and *blue* (hence, the result of mixing them) colorimetrically.

When the exact chromaticities of red, green, and blue primaries are defined, we have a *color space*. There are several color spaces, such as CIERGB, CIEXYZ, or sRGB. CIERGB and CIEXYZ are the first formal color spaces defined by the CIE in 1931. Since display devices can only generate non-negative primaries, and an adequate amount of luminance is required, there is, in practice, a limitiation on the gamut of colors that can be reproduced on a given device. Color characteristics of a device can be specified by its International Color Consortium (ICC) profile.

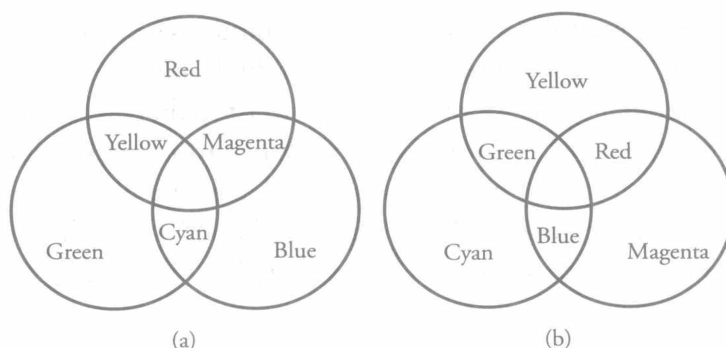



**Figure 2.2** Color spaces: (a) additive color space and (b) subtractive color space.

### Color Management

Color management must be employed to generate the exact same color on different devices, where the device-dependent color values of the input device, given its ICC profile, is first mapped to a standard device-independent color space, sometimes called the Profile Connection Space (PCS), such as CIEXYZ. They are then mapped to the device-dependent color values of the output device given the ICC profile of the output device. Hence, an ICC profile is essentially a mapping from a device color space to the PCS and from the PCS to a device color space. Suppose we have particular RGB and CMYK devices and want to convert the RGB values to CMYK. The first step is to obtain the ICC profiles of concerned devices. To perform the conversion, each (R, G, B) triplet is first converted to the PCS using the ICC profile of the RGB device. Then, the PCS is converted to the C, M, Y, and K values using the profile of the second device.

Color management may be side-stepped by calibrating all devices to a common standard color space, such as sRGB, which was developed by HP and Microsoft in 1996. sRGB uses the color primaries defined by the ITU-R recommendation BT.709, which standardizes the format of high-definition television. When such a calibration is done well, no color translations are needed to get all devices to handle colors consistently. Avoiding the complexity of color management was one of the goals in developing sRGB [IEC 00].

## 2.1.2 Contrast Sensitivity

Contrast can be defined as the difference between the luminance of a region and its background. The human visual system is more sensitive to contrast than absolute

luminance; hence, we can perceive the world around us similarly regardless of changes in illumination. Since most images are viewed by humans, it is important to understand how the human visual system senses contrast so that algorithms can be designed to preserve the more visible information and discard the less visible ones. Contrast-sensitivity mechanisms of human vision also determine which compression or processing artifacts we see and which we don't. The ability of the eye to discriminate between changes in intensity at a given intensity level is quantified by Weber's law.

### Weber's Law

Weber's law states that smaller intensity differences are more visible on a darker background and can be quantified as

$$\frac{\Delta I}{I} = c \text{ (constant), for } I > 0 \tag{2.3}$$

where $\Delta I$ is the just noticeable diference (JND) [Gon 07]. Eqn. (2.3) states that the JND grows proportional to the intensity level $I$. Note that $I = 0$ denotes the darkest intensity, while $I = 255$ is the brightest. The value of $c$ is empirically found to be around 0.02. The experimental set-up to measure the JND is shown in Figure 2.3(a). The rods and cones comply with Weber's law above -2.6 log candelas (cd)/m2 (moonlight) and 2 log cd/m2 (indoor) luminance levels, respectively [Fer 01].

### Brightness Adaptation

The human eye can adapt to different illumination/intensity levels [Fer 01]. It has been observed that when the background-intensity level the observer has adapted to is different from $I$, the observer's intensity resolution ability decreases. That is, when $I_0$ is different from $I$, as shown in Figure 2.3(b), the JND $\Delta I$ increases relative to the case $I_0 = I$. Furthermore, the *simultanenous contrast effect* illustrates that humans perceive the brightness of a square with constant intensity differently as the intensity of the background varies from light to dark [Gon 07].

It is also well-known that the human visual system undershoots and overshoots around the boundary of step transitions in intensity as demonstrated by the *Mach band effect* [Gon 07].

### Visual Masking

Visual masking refers to a nonlinear phenomenon experimentally observed in the human visual system when two or more visual stimuli that are closely coupled in space or time are presented to a viewer. The action of one visual stimulus on the visibility of another is called masking. The effect of masking may be a decrease in

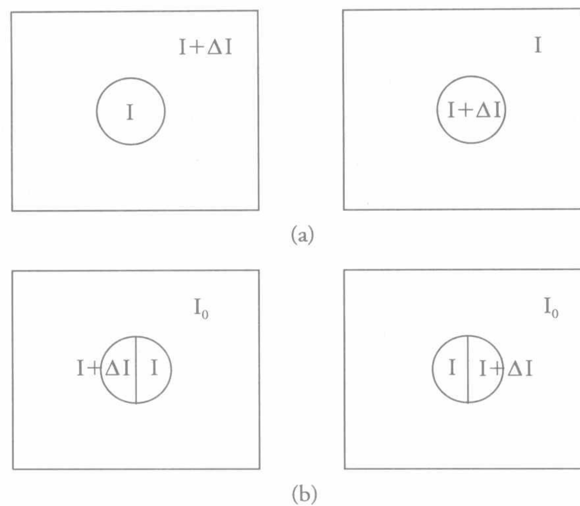



**Figure 2.3** Illustration of (a) the just noticeable difference and (b) brightness adaptation.

brightness or failure to detect the target or some details, e.g., texture. Visual masking can be studied under two cases: *spatial masking* and *temporal masking*.

#### Spatial Masking

Spatial masking is observed when a viewer is presented with a superposition of a target pattern and mask (background) image [Fer 01]. The effect states that the visibility of the target pattern is lower when the background is spatially busy. Spatial busyness measures include local image variance or textureness. Spatial masking implies that visibility of noise or artifact patterns is lower in spatially busy areas of an image as compared to spatially uniform image areas.

#### Temporal Masking

Temporal masking is observed when two stimuli are presented sequentially [Bre 07]. Salient local changes in luminance, hue, shape, or size may become undetectable in the presence of large coherent object motion [Suc 11]. Considering video frames as a sequence of stimuli, fast-moving objects and scene cuts can trigger a temporal-masking effect.

### 2.1.3 Spatio-Temporal Frequency Response

An understanding of the response of the human visual system to spatial and temporal frequencies is important to determine video-system design parameters and video-compression parameters, since frequencies that are invisible to the human eye are irrelevant.

### Spatial-Frequency Response

Spatial frequencies are related to how still (static) image patterns vary in the horizontal and vertical directions in the spatial plane. The spatial-frequency response of the human eye varies with the viewing distance; i.e., the closer we get to the screen the better we can see details. In order to specify the spatial frequency independent of the viewing distance, spatial frequency (in cycles/distance) must be normalized by the viewing distance $d$, which can be done by defining the viewing angle $\theta$ as shown in Figure 2.4(a).

Let $w$ denote the picture width. If $w/2 \ll d$, then $\frac{\theta}{2} \approx \sin\frac{\theta}{2} = \frac{w/2}{d}$, considering the right triangle formed by the viewer location, an end of the picture, and the middle of the picture. Hence,

$$\theta \approx \frac{w}{d}(\text{radians}) = \frac{180w}{\pi d}(\text{degrees}) \tag{2.4}$$

Let $f_w$ denote the number of cycles per picture width, then the normalized horizontal spatial frequency (i.e., number of cycles per viewing degree) $f_\theta$ is given by

$$f_\theta = \frac{f_w}{\theta} = \frac{f_w d}{w}(\text{cycles / radian}) = \frac{\pi\ d\ f_w}{180\ w}(\text{cycles / degree}) \tag{2.5}$$

The normalized vertical spatial frequency can be defined similarly in the units of cycles/degree. As we move away from the screen $d$ increases, and the same number of cycles per picture width $f_w$ appears as a larger frequency $f_\theta$ per viewing degree. Since the human eye has reduced contrast sensitivity at higher frequencies, the same pattern is more difficult to see from a larger distance $d$. The horizontal and vertical resolution (number of pixels and lines) of a TV has been determined such that horizontal and vertical sampling frequencies are twice the highest frequency we can see (according to the Nyquist sampling theorem), assuming a fixed value for the ratio $d/w$—i.e., viewing distance over picture width. Given a fixed viewing distance, clearly we need more video resolution (pixels and lines) as picture (screen) size increases to experience the same video quality.

Figure 2.4(b) shows the spatial-frequency response, which varies by the average luminance level, of the eye for both the luminance and chrominance components of still images. We see that the spatial-frequency response of the eye, in general, has low-pass/band-pass characteristics, and our eyes are more sensitive to higher frequency patterns in the luminance components compared with those in the chrominance components. The latter observation is the basis of the conversion from RGB to the luminance-chrominance space for color image processing and the reason we subsample the two chrominance components in color image/video compression.

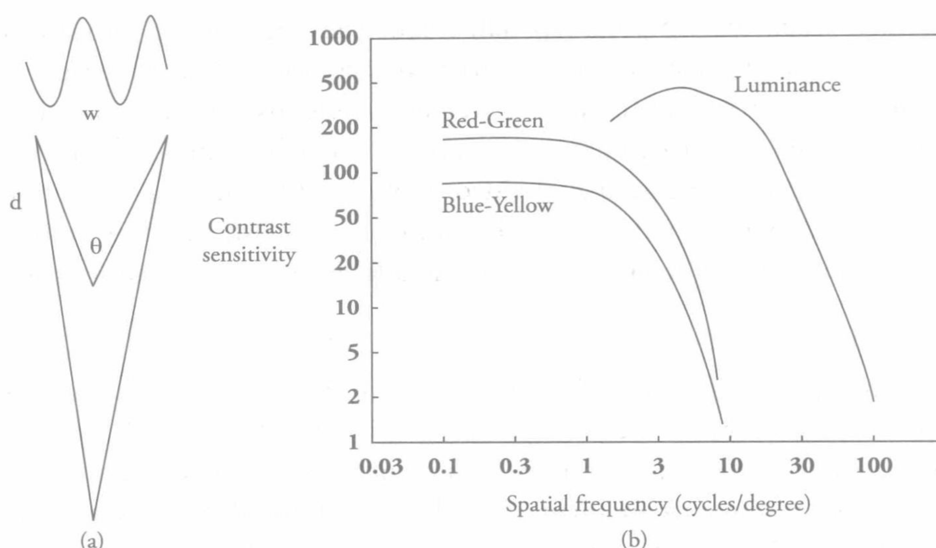



**Figure 2.4** Spatial frequency and spatial response: (a) viewing angle and (b) spatial-frequency response of the human eye [Mul 85].

### Temporal-Frequency Response

Video is displayed as a sequence of still frames. The frame rate is measured in terms of the number of pictures (frames) displayed per second or Hertz (Hz). The frame rates for cinema, television, and computer monitors have been determined according to the temporal-frequency response of our eyes. The human eye has lower sensitivity to higher temporal frequencies due to temporal integration of incoming light into the retina, which is also known as vision persistence. It is well known that the integration period is inversely proportional to the incoming light intensity. Therefore, we can see higher temporal frequencies on brighter screens. Psycho-visual experiments indicate the human eye cannot perceive flicker if the refresh rate of the display (temporal frequency) is more than 50 times per second for TV screens. Therefore, the frame rate for TV is set at 50-60 Hz, while the frame rate for brighter computer monitors is 72 Hz or higher, since the brighter the screen the higher the critical flicker frequency.

### Interaction Between Spatial- and Temporal-Frequency Response

Video exhibits both spatial and temporal variations, and spatial- and temporal-frequency responses of the eye are not mutually independent. Hence, we need to understand the spatio-temporal frequency response of the eye. The effects of changing average luminance on the contrast sensitivity for different combinations of spatial and temporal frequencies have been investigated [Nes 67]. Psycho-visual experiments

indicate that when the temporal (spatial) frequencies are close to zero, the spatial (temporal) frequency response has bandpass characteristics. At high temporal (spatial) frequencies, the spatial (temporal) frequency response has low-pass characteristics with smaller cut-off frequency as temporal (spatial) frequency increases. This implies that we can exchange spatial video resolution for temporal resolution, and vice versa. Hence, when a video has high motion (moves fast), the eyes cannot sense high spatial frequencies (details) well if we exclude the effect of eye movements.

**Eye Movements**

The human eye is similar to a sphere that is free to move like a ball in a socket. If we look at a nearby object, the two eyes turn in; if we look to the left, the right eye turns in and the left eye turns out; if we look up or down, both eyes turn up or down together. These movements are directed by the brain [Hub 88]. There are two main types of gaze-shifting eye movements, saccadic and smooth pursuit, that affect the spatial- and spatio-temporal frequency response of the eye. Saccades are rapid movements of the eyes while scanning a visual scene. "Saccadic eye movements" enable us to scan a greater area of the visual scene with the high-resolution fovea of the eye. On the other hand, "smooth pursuit" refers to movements of the eye while tracking a moving object, so that a moving image remains nearly static on the high-resolution fovea. Obviously, smooth pursuit eye movements affect the spatio-temporal frequency response of the eye. This effect can be modeled by tracking eye movements of the viewer and motion compensating the contrast sensitivity function accordingly.

### 2.1.4 Stereo/Depth Perception

Stereoscopy creates the illusion of 3D depth from two 2D images, a left and a right image that we should view with our left and right eyes. The horizontal distance between the eyes (called *interpupilar distance*) of an average human is 6.5 cm. The difference between the left and right retinal images is called *binocular disparity*. Our brain deducts depth information from this binocular disparity. 3D display technologies that enable viewing of right and left images with our right and left eyes, respectively, are discussed in Section 2.3.1.

**Accomodation, Vergence, and Visual Discomfort**

In human stereo vision, there are two oculomotor mechanisms, accommodation (where we focus) and vergence (where we look), which are reflex eye movements. Accommodation is the process by which the eye changes optical focus to maintain a clear image of an object as its distance from the eye varies. Vergence or convergence

are the movements of both eyes to make sure the image of the object being looked at falls on the corresponding spot on both retinas. In real 3D vision, accommodation and vergence distances are the same. However, in flat 3D displays both left and right images are displayed on the plane of the screen, which determines the accommodation distance, while we look and perceive 3D objects at a different distance (usually closer to us), which is the vergence distance. This difference between accommodation and vergence distances may cause serious discomfort if it is greater than some tolerable amount. The depth of an object in the scene is determined by the disparity value, which is the displacement of a feature point between the right and left views. The depth, hence the difference between accommodation and vergence distances, can be controlled by 3D-video (disparity) processing at the content preparation stage to provide a comfortable 3D viewing experience.

Another cause of viewing discomfort is the cross-talk between the left and right views, which may cause ghosting and blurring. Cross-talk may result from imperfections in polarizing filters (passive glasses) or synchronization errors (active shutters), but it is more prominent in auto-stereoscopic displays where the optics may not completely prevent cross-talk between the left and right views.

**Binocular Rivalry/Suppression Theory**

Binocular rivalry is a visual perception phenomenon that is observed when different images are presented to right and left eyes [Wad 96]. When the quality difference between the right and left views are small, according to the suppression theory of stereo vision, the human eye can tolerate absence of high-frequency content in one of the views; therefore, two views can be represented at unequal spatial resolutions or quality. This effect has lead to asymmetric stereo-video coding, where only the dominant view is encoded with high fidelity (bitrate). The results have shown that perceived 3D-video quality of such asymmetric processed stereo pairs is similar to that of symmetrically encoded sequences at higher total bitrate. They also observe that scaling (zoom in/out) one or both views of a stereoscopic test sequence does not affect depth perception. We note that these results have been confirmed on short test sequences. It is not known whether asymmetric view resolution or quality would cause viewing discomfort over longer videos with increased period of viewing.

## 2.2 Digital Video

We have experienced a digital media revolution in the last couple of decades. TV and cinema have gone all-digital and high-definition, and most movies and some TV

broadcasts are now in 3D format. High-definition digital video has landed on laptops, tablets, and cellular phones with high-quality media streaming over the Internet. Apart from the more robust form of the digital signal, the main advantage of digital representation and transmission is that they make it easier to provide a diverse range of services over the same network. Digital video brings broadcasting, cinema, computers, and communications industries together in a truly revolutionary manner, where telephone, cable TV, and Internet service providers have become fierce competitors. A single device can serve as a personal computer, a high-definition TV, and a videophone. We can now capture live video on a mobile device, apply digital processing on a laptop or tablet, and/or print still frames at a local printer. Other applications of digital video include medical imaging, surveillance for military and law enforcement, and intelligent highway systems.

### 2.2.1 Spatial Resolution and Frame Rate

Digital-video systems use component color representation. Digital color cameras provide individual RGB component outputs. Component color video avoids the artifacts that result from analog composite encoding. In digital video, there is no need for blanking or sync pulses, since it is clear where a new line starts given the number of pixels per line.

The horizontal and vertical resolution of digital video is related to the pixel sampling density, i.e., the number of pixels per unit distance. The number of pixels per line and the number of lines per frame is used to classify video as standard, high, or ultra-high defnition, as depicted in Figure 2.5. In low-resolution digital video, pixel - lation (aliasing) artifact arises due to lack of sufficient spatial resolution. It manifests itself as jagged edges resulting from individual pixels becoming visible. The visibility of pixellation artifacts varies with the size of the display and the viewing distance. This is quite different from analog video where the lack of spatial-resolution results in blurring of image in the respective direction.

The frame/field rate is typically 50/60 Hz, although some displays use frame interpolation to display at 100/120, 200 or even 400 Hz. The notation 50i (or 60i) indicates interlaced video with 50 (60) fields/sec, which corresponds to 25 (30) pictures/sec obtained by weaving the two fields together. On the other hand, 50p (60p) denotes 50 (60) full progressive frames/sec.

The arrangement of pixels and lines in a contiguous region of the memory is called a bitmap. There are five key parameters of a bitmap: the starting address in the memory, the number of pixels per line, the pitch value, the number of lines, and the number of bits per pixel. The pitch value specifies the distance in memory

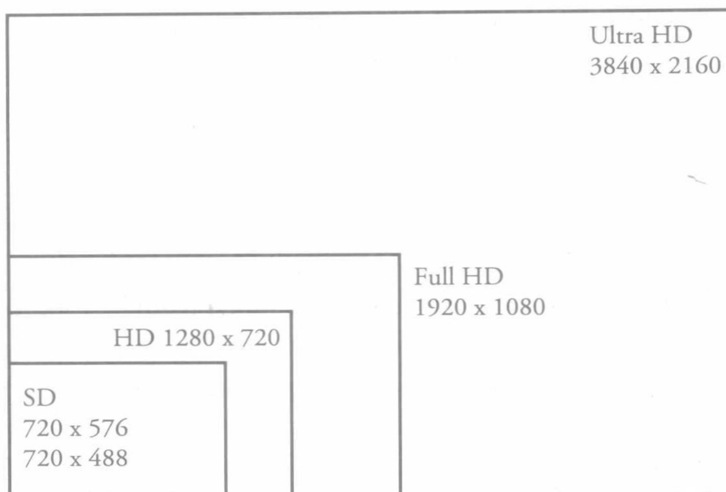



**Figure 2.5** Digital-video spatial-resolution formats.

from the start of one line to the next. The most common use of pitch different from the number of pixels per line is to set pitch to the next highest power of 2, which may help certain applications run faster. Also, when dealing with interlaced inputs, setting the pitch to double the number of pixels per line facilitates writing lines from each field alternately in memory. This will form a "weaved frame" in a contiguous region of the memory.

## 2.2.2 Color, Dynamic Range, and Bit-Depth

This section addresses color representation, dynamic range, and bit-depth in digital images/video.

### Color Capture and Display

Color cameras can be the three-sensor type or single-sensor type. Three-sensor cameras capture R, G, and B components using different CCD panels, using an optical beam splitter; however, they may suffer from synchronicity problems and high cost, while single-sensor cameras often have to compromise spatial resolution. This is because a color filter array is used so that each CCD element captures one of R, G, or B pixels in some periodic pattern. A commonly used color filter pattern is the Bayer array, shown in Figure 2.6, where two out of every four pixels are green, one is red, and one is blue, since green signal contributes the most to the luminance channel. The missing pixel values in each color channel are computed by linear or adaptive

interpolation filters, which may result in some aliasing artifacts. Similar color filter array patterns are also employed in LCD/LED displays, where the human eye performs low-pass filtering to perceive a full-colored image.

### Dynamic Range

The dynamic range of a capture device (e.g., a camera or scanner) or a display device is the ratio between the maximum and minimum light intensities that can be represented. The luminance levels in the environment range from $-4$ log cd/m$^2$ (starlight) to 6 log cd/m$^2$ (sun light); i.e., the dynamic range is about 10 log units [Fer 01]. The human eye has complex fast and slow adaptation schemes to cope with this large dynamic range. However, a typical imaging device (camera or display) has a maximum dynamic range of 300:1, which corresponds to 2.5 log units. Hence, our ability to capture and display a foreground object subject to strong backlighting with proper contrast is limited. High dynamic range (HDR) imaging aims to remedy this problem.

#### *HDR Image Capture*

HDR image capture with a standard dynamic range camera requires taking a sequence of pictures at different exposure levels, where raw pixel exposure data (linear in exposure time) are combined by weighted averaging to obtain a single HDR image [Gra 10]. There are two possible ways to display HDR images: i) employ

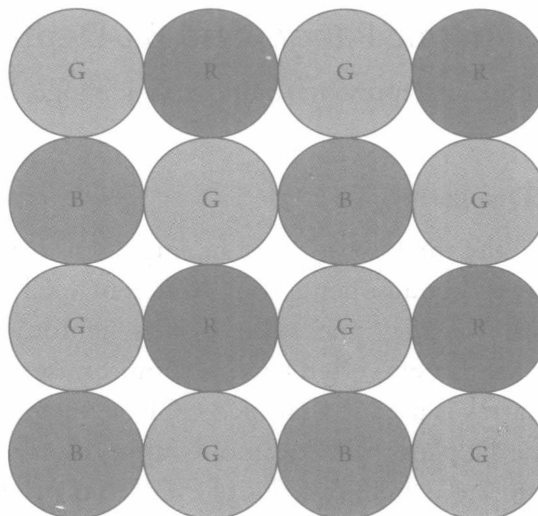



**Figure 2.6** Bayer color-filter array pattern.

new higher dynamic range display technologies, or ii) employ local tone-mapping algorithms for dynamic range compression (see Chapter 3) to better render details in bright or dark areas on a standard display [Rei 07].

*HDR Displays*

Recently, new display technologies that are capable of up to 50,000:1 or 4.7 log units dynamic range with maximum intensity 8500 cd/m$^2$, compared to standard displays with contrast ratio 2 log units and maximum intensity 300 cd/m$^2$, have been proposed [See 04]. This high dynamic range matches the human eye's short time-scale (fast) adaptation capability well, which enables our eyes to capture approximately 5 log units of dynamic range at the same time.

**Bit-Depth**

Image-intensity values at each sample are quantized for a finite-precision representation. Today, each color component signal is typically represented with 8 bits per pixel, which can capture 255:1 dynamic range for a total of 24 bits/pixel and $2^{24}$ distinct colors to avoid "contouring artifacts." Contouring results in slowly varying regions of image intensity due to insufficient bit resolution. Some applications, such as medical imaging and post-production editing of motion pictures may require 10, 12, or more bits/pixel/color. In high dynamic range imaging, 16 bits/pixel/color is required to capture a 50,000:1 dynamic range, which is now supported in JPEG.

Digital video requires much higher data rates and transmission bandwidths as compared to digital audio. CD-quality digital audio is represented with 16 bits/sample, and the required sampling rate is 44 kHz. Thus, the resulting data rate is approximately 700 kbits/sec (kbps). This is multiplied by 2 for stereo audio. In comparison, a high-definition TV signal has 1920 pixels/line and 1080 lines for each luminance frame, and 960 pixels/line and 540 lines for each chrominance frame. Since we have 25 frames/sec and 8 bits/pixel/color, the resulting data rate exceeds 700 Mbps, which testifies to the statement that a picture is worth 1000 words! Thus, the feasibility of digital video is dependent on image-compression technology.

### 2.2.3 Color Image Processing

Color images/video are captured and displayed in the RGB format. However, they are often converted to an intermediate representation for efficient compression and processing. We review the luminance-chrominance (for compression and filtering) and the normalized RGB and hue-saturation-intensity (HSI) (for color-specific processing) representations in the following.

### Luminance-Chrominance

The luminance-chrominance color model was used to develop an analog color TV transmission system that is backwards compatible with the legacy analog black and white TV systems. The luminance component, denoted by Y, corresponds to the gray-level representation of video, while the two chrominance components, denoted by U and V for analog video or Cr and Cb for digital video, represent the deviation of color from the gray level on blue–yellow and red–cyan axes. It has been observed that the human visual system is less sensitive to variations (higher frequencies) in chrominance components (see Figure 2.4(b)). This has resulted in the subsampled chrominance formats, such as 4:2:2 and 4:2:0. In the 4:2.2 format, the chrominance components are subsampled only in the horizontal direction, while in 4:2:0 they are subsampled in both directions as illustrated in Figure 2.7. Te luminance-chrominance representation offers higher compression efficiency, compared to the RGB representation due to this subsampling.

ITU-R BT.709 defines the conversion between RGB and YCrCb representations as:

$$\begin{aligned} Y &= 0.299\ R + 0.587\ G + 0.114\ B \\ Cr &= 0.499\ R - 0.418\ G - 0.0813\ B + 128 \\ Cb &= -0.169\ R - 0.331\ G + 0.499\ B + 128 \end{aligned} \tag{2.6}$$

which states that the human visual system perceives the contribution of R-G-B to image intensity approximately with a 3-6-1 ratio, i.e., red is weighted by 0.3, green by 0.6 and blue by 0.1.

The inverse conversion is given by

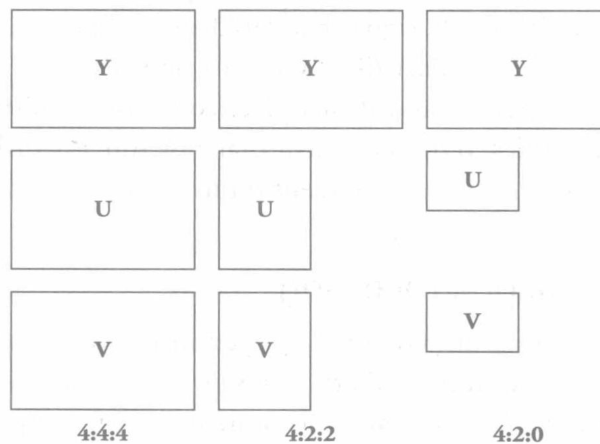



Figure 2.7 Chrominance subsampling formats: (a) no subsampling; (b) 4:2:2; (c) 4:2:0 format.

$$
\begin{aligned}
R &= Y + 1.402\,(Cr - 128) \\
G &= Y - 0.714\,(Cr - 128) - 0.344\,(Cb - 128) \\
B &= Y + 1.772\,(Cb - 128)
\end{aligned}
\tag{2.7}
$$

The resulting R, G, and B values must be truncated to the range (0, 255) if they fall outside. We note that Y-Cr-Cb is not a color space. It is a way of encoding the RGB information, and actual colors displayed depends on the specific RGB space used.

A common practice in color image processing, such as edge detection, enhancement, denoising, restoration, etc., in the luminance-chrominance domain is to process only the luminance (Y) component of the image. There are two main reasons for this: i) processing R, G, and B components independently may alter the color balance of the image, and ii) the human visual system is not very sensitive to high frequencies in the chrominance components. Therefore, we first convert a color image into Y-Cr-Cb color space, then perform image enhancement, denoising, restoration, etc., on the Y channel only. We then transform the processed Y channel and unprocessed Cr and Cb channels back to the R-G-B domain for display.

### Normalized rgb

Normalized rgb components aim to reduce the dependency of color represented by the RGB values on image brightness. They are defined by

$$
\begin{aligned}
r &= R/(R+G+B) \\
g &= G/(R+G+B) \\
b &= B/(R+G+B)
\end{aligned}
\tag{2.8}
$$

The normalized $r, g, b$ values are always within the range 0 to 1, and

$$
r + g + b = 1 \tag{2.9}
$$

Hence, they can be specified by any two components, typically by $(r, g)$ and the third component can be obtained from Eqn. (2.9). The normalized rgb domain is often used in color-based object detection, such as skin-color or face detection.

**Example.** We demonstrate how the normalized rgb domain helps to detect similar colors independent of brightness by means of an example: Let's assume we have two pixels with (R, G, B) values (230, 180, 50) and (115, 90, 25). It is clear that the second pixel is half as bright as the first, which may be because it is in a shadow. In the normalized rgb, both pixels

are represented by $r = 0.50$, $g = 0.39$, and $b = 0.11$. Hence, it is apparent that they represent the same color after correcting for brightness difference by the normalization.

**Hue-Saturation-Intensity (HSI)**

Color features that best correlate with human perception of color are hue, saturation, and intensity. Hue relates to the dominant wavelength, saturation relates to the spread of power about this wavelength (purity of the color), and intensity relates to the perceived luminance (similar to the Y channel). There is a family of color spaces that specify colors in terms of hue, saturation, and intensity, known as HSI spaces. Conversion to HSI where each component is in the range [0,1] can be performed from the scaled RGB, where each component is divided by 255 so they are in the range [0,1]. The HSI space specifies color in cylindrical coordinates and conversion formulas (2.10) are nonlinear [Gon 07].

$$H = \begin{cases} \theta & \text{if } B \leq G \\ 360 - \theta & \text{if } B > G \end{cases} \quad \text{where } \theta = \arccos\left\{ \frac{\frac{1}{2}[(R-G)+(R-B)]}{\sqrt{(R-G)^2 + (R-B)(G-B)}} \right\}$$

$$S = 1 - \frac{3\min\{R, G, B\}}{R+G+B} \tag{2.10}$$

$$I = \frac{R+G+B}{3}$$

Note that HSI is not a perceptually uniform color space, i.e., equal perturbations in the component values do not result in perceptually equal color variations across the range of component values. The CIE has also standardized some perceptually uniform color spaces, such as L*, u*, v* and L*, a*, b* (CIELAB).

### 2.2.4 Digital-Video Standards

Exchange of digital video between different products, devices, and applications requires digital-video standards. We can group digital-video standards as video-format (resolution) standards, video-interface standards, and image/video compression standards. In the early days of analog TV, cinema (film), and cameras (cassette), the computer, TV, and consumer electronics industries established different display resolutions and scanning standards. Because digital video has brought cinema, TV, consumer electronics, and computer industries ever closer, standardization across industries has started. This section introduces recent standards and standardization efforts.

### Video-Format Standards

Historically, standardization of digital-video formats originated from different sources: ITU-R driven by the TV industry, SMPTE driven by the motion picture industry, and computer/consumer electronics associations.

Digital video was in use in broadcast TV studios even in the days of analog TV, where editing and special effects were performed on digitized video because it is easier to manipulate digital images. Working with digital video avoids artifacts that would otherwise be caused by repeated analog recording of video on tapes during various production stages. Digitization of analog video has also been needed for conversion between different analog standards, such as from PAL to NTSC, and vice versa. ITU-R (formerly CCIR) Recommendation BT.601 defines a *standard definition TV* (SDTV) digital-video format for 525-line and 625-line TV systems, also known as digital studio standard, which is originally intended to digitize analog TV signals to permit digital post-processing as well as international exchange of programs. This recommendation is based on component video with one luminance (Y) and two chrominance (Cr and Cb) signals. The sampling frequency for analog-to-digital (A/D) conversion is selected to be an integer multiple of the horizontal sweep frequencies (line rates) $f_{h,525} = 525 = 29.97 = 15{,}734$ and $f_{h,625} = 625 \times 25 = 15{,}625$ in both 525- and 625-line systems. Thus, for the luminance

$$f_{s,lum} = 858\ f_{h,525} = 864\ f_{h,625} = 13.5\ \text{MHz}$$

i.e., 525 and 625 line systems have 858 and 864 samples/line, respectively, and for chrominance

$$f_{s,chr} = f_{s,lum}/2 = 6.75\ \text{MHz}$$

ITU-R BT.601 standards for both 525- and 625-line SDTV systems employ interlaced scan, where the raw data rate is 165.9 Mbps. The parameters of both formats are shown in Table 2.1. Historically, interlaced SDTV was displayed on analog cathode ray tube (CRT) monitors, which employ interlaced scanning at 50/60 Hz. Today, flat-panel displays and projectors can display video at 100/120 Hz interlace or progressive mode, which requires scan-rate conversion and de-interlacing of the 50i/60i ITU-R BT.601 [ITU 11] broadcast signals.

Recognizing that the resolution of SDTV is well behind today's technology, a new *high-definition TV* (HDTV) standard, ITU-R BT.709-5 [ITU 02], which doubles the resolution of SDTV in both horizontal and vertical directions, has been approved

with three picture formats: 720p, 1080i, and 1080p. Table 2.1 shows their parameters. Today broadcasters use either 720p/50/60 (called HD) or 1080i/25/29.97 (called FullHD). There are no broadcasts in 1080p format at this time. Note that many 1080i/25 broadcasts use horizontal sub-sampling to 1440 pixels/line to save bitrate. 720p/50 format has full temporal resolution 50 progressive frames per second (with 720 lines). Note that most international HDTV events are captured in either 1080i/25 or 1080i/29.97 (for 60 Hz countries) and presenting 1080i/29.97 in 50 Hz countries or vice versa requires scan rate conversion. For 1080i/25 content, 720p/50 broadcasters will need to de-interlace the signal before transmission, and for 1080i/29.97 content, both de-interlacing and frame-rate conversion is required. Furthermore, newer 1920 × 1080 progressive scan consumer displays require up-scaling 1280 × 720 pixel HD broadcast and 1440 × 1080i/25 sub-sampled FullHD broadcasts.

In the computer and consumer electronics industry, standards for video-display resolutions are set by a consortia of organizations such as Video Electronics Standards Association (VESA) and Consumer Electronics Association (CEA). The display standards can be grouped as Video Graphics Array (VGA) and its variants and Extended Graphics Array (XGA) and its variants. The favorite aspect ratio of the display industry has shifted from the earlier 4:3 to 16:10 and 16:9. Some of these standards are shown in Table 2.2. The refresh rate was an important parameter for CRT monitors. Since activated LCD pixels do not flash on/off between frames, LCD monitors do not exhibit refresh-induced flicker. The only part of an LCD monitor that can produce CRT-like flicker is its backlight, which typically operates at 200 Hz.

Recently, standardization across TV, consumer electronics, and computer industries has started, resulting in the so-called convergence enabled by digital video. For example, some laptops and cellular phones now feature 1920 × 1080 progressive

**Table 2.1** ITU-R TV Broadcast Standards

| Standard | Pixels | Lines | Interlace/Progressive, Picture Rate | Aspect Ratio |
|---|---|---|---|---|
| BT.601-7 480i | 720 | 486 | 2:1 Interlace, 30 Hz (60 fields/s) | 4:3, 16:9 |
| BT.601-7 576i | 720 | 576 | 2:1 Interlace, 25 Hz (50 fields/s) | 4:3, 16:9 |
| BT.709-5 720p | 1280 | 720 | Progressive, 50 Hz, 60 Hz | 16:9 |
| BT.709-5 1080i | 1920 | 1080 | 2:1 Interlace, 25 Hz, 30 Hz | 16:9 |
| BT.709-5 1080p | 1920 | 1080 | Progressive | 16:9 |
| BT.2020 2160p | 3840 | 2160 | Progressive | 16:9 |
| BT.2020 4320p | 7680 | 4320 | Progressive | 16:9 |

mode, which is a format jointly supported by TV, consumer electronics, and computer industries.

*Ultra-high definition television* (UHDTV) is the most recent standard proposed by NHK Japan and approved as ITU-R BT.2020 [ITU 12]. It supports the 4K (2160p) and 8K (4320p) digital-video formats shown in Table 2.1. The Consumer Electronics Association announced that "ultra high-definition" or "ultra HD" or "UHD" would be used for displays that have an aspect ratio of at least 16:9 and at least one digital input capable of carrying and presenting native video at a minimum resolution of $3{,}840 \times 2{,}160$ pixels. The ultra-HD format is very similar to 4K digital cinema format (see Section 2.4.2) and may become an across industries standard in the near future.

### Video-Interface Standards

Digital-video interface standards enable exchange of uncompressed video between various consumer electronics devices, including digital TV monitors, computer monitors, blu-ray devices, and video projectors over cable. Two such standards are Digital Visual Interface (DVI) and High-Definition Multimedia Interface (HDMI). HDMI is the most popular interface that enables transfer of video and audio on a single cable. It is backward compatible with DVI-D or DVI-I. HDMI 1.4 and higher support 2160p digital cinema and 3D stereo transfer.

### Image- and Video-Compression Standards

Various digital-video applications, e.g., SDTV, HDTV, 3DTV, video on demand,

**Table 2.2** Display Standards

| Standard | Pixels | Lines | Aspect Ratio |
|---|---|---|---|
| VGA | 640 | 480 | 4:3 |
| WSVGA | 1024 | 576 | 16:9 |
| XGA | 1024 | 768 | 4:3 |
| WXGA | 1366 | 768 | 16:9 |
| SXGA | 1280 | 1024 | 5:4 |
| UXGA | 1600 | 1200 | 4:3 |
| FHD | 1920 | 1080 | 16:9 |
| WUXGA | 1920 | 1200 | 16:10 |
| HXGA | 4096 | 3072 | 4:3 |
| WQUXGA | 3840 | 2400 | 16:10 |
| WHUXGA | 7680 | 4800 | 16:10 |

interactive games, and videoconferencing, reach potential users over either broadcast channels or the Internet. Digital cinema content must be transmitted to movie theatres over satellite links or must be shipped in harddisks. Raw (uncompressed) data rates for digital video are prohibitive, since uncompressed broadcast HDTV requires over 700 Mbits/s and 2K digital cinema data exceeds 5 Gbits/sec in uncompressed form. Hence, digital video must be stored and transmitted in compressed form, which leads to compression standards.

Video compression is a key enabling technology for digital video. Standardization of image and video compression is required to ensure compatibility of digital-video products and hardware by different vendors. As a result, several video-compression standards have been developed, and work for even more efficient compression is ongoing. Major standards for image and video compression are listed in Table 2.3.

Historically, standardization in digital-image communication started with the ITU-T (formerly CCITT) digital fax standards. The ITU-T Recommendation T.4 using 1D coding for digital fax transmission was ratified in 1980. Later, a more efficient 2D compression technique was added as an option to the ITU-T recommendation T.30 and ISO JBIG was developed to fix some of the problems with the ITU-T Group 3 and 4 codes, mainly in the transmission of half-tone images.

JPEG was the first color still-image compression standard. It has also found some use in frame-by-frame video compression, called motion JPEG, mostly because of its wide availability in hardware. Later JPEG2000 was developed as a more efficient alternative especially at low bit rates. However, it has mainly found use in the digital cinema standards.

The first commercially successful video-compression standard was MPEG-1 for video storage on CD, which is now obsolete. MPEG-2 was developed for compression of SDTV and HDTV as well as video storage in DVD and was the enabling

**Table 2.3** International Standards for Image/Video Compression

| Standard | Application |
|---|---|
| ITU-T (formerly CCITT) G3/G4 | FAX, Binary images |
| ISO JBIG | Binary/halftone, gray-scale images |
| ISO JPEG | Still images |
| ISO JPEG2000 | Digital cinema |
| ISO MPEG2 | Digital video, SDTV, HDTV |
| ISO MPEG4 AVC/ITU-T H.264 | Digital video |
| ISO HEVC/ ITU-T H.265 | HD video, HDTV, UHDTV |

technology of digital TV. MPEG-4 AVC and HEVC were later developed as more efficient compression standards especially for HDTV and UHDTV as well as video on blu-ray discs. We discuss image- and video-compression technologies and standards in detail in Chapter 7 and Chapter 8, respectively.

## 2.3 3D Video

3D cinema has gained wide acceptance in theatres as many movies are now produced in 3D. Flat-panel 3DTV has also been positively received by consumers for watching sports broadcasts and blu-ray movies. Current 3D-video displays are stereoscopic and are viewed by special glasses. Stereo-video formats can be classified as frame-compatible (mainly for broadcast TV) and full-resolution (sequential) formats. Alternatively, multi-view and super multi-view 3D-video displays are currently being developed for autostereoscopic viewing. Multi-view video formats without accompanying depth information require extremely high data rates. Multi-view-plus-depth representation and compression are often preferred for efficient storage and transmission of multi-view video as the number of views increases. There are also volumetric, holoscopic (integral imaging), and holographic 3D-video formats, which are mostly considered as futuristic at this time.

The main technical obstacles for 3DTV and video to achieve much wider acceptance at home are: i) developing affordable, free-viewing natural 3D display technologies with high spatial, angular, and depth resolution, and ii) capturing and producing 3D content in a format that is suitable for these display technologies. We discuss 3D display technologies and 3D-video formats in more detail below.

### 2.3.1 3D-Display Technologies

A 3D display should ideally reproduce a light field that is an indistinguishable copy of the actual 3D scene. However, this is a rather difficult task to achieve with today's technology due to very large amounts of data that needs to be captured, processed, and stored/transmitted. Hence, current 3D displays can only reproduce a limited set of 3D visual cues instead of the entire light field; namely, they reproduce:

- Binocular depth – Binocular disparity in a stereo pair provides relative depth cue. 3D displays that present only two views, such as stereo TV and digital cinema, can only provide binocular depth cue.
- Head-motion parallax – Viewers expect to see a scene or objects from a slightly different perspective when they move their head. Multi-view, light-field, or vol-

umetric displays can provide head-motion parallax, although most displays can provide only limited parallax, such as only horizontal parallax.

We can broadly classify 3D display technologies as multiple-image (stereoscopic and auto-stereoscopic), light-field, and volumetric displays, as summarized in Figure 2.8. *Multiple-image displays* present two or more images of a scene by some multiplexing of color sub-pixels on a planar screen such that the right and left eyes see two separate images with binocular disparity, and rely upon the brain to fuse the two images to create the sensation of 3D. *Light-field displays* present light rays as if they are originating from a real 3D object/scene using various technologies such that each pixel of the display can emit multiple light rays with different color, intensity, and directions, as opposed to multiplexing pixels among different views. *Volumetric displays* aim to reconstruct a visual representation of an object/scene using voxels with three physical dimensions via emission, scattering, or relaying of light from a well-defined region in the physical $(x_1, x_2, x_3)$ space, as opposed to displaying light rays emitted from a planar screen.

### *Multiple-Image Displays*

Multiple-image displays can be classified as those that require glasses (stereoscopic) and those that don't (auto-stereoscopic).

Stereoscopic displays present two views with binocular disparity, one for the left and one for the right eye, from a single viewpoint. Glasses are required to ensure that only the right eye sees the right view and the left eye sees the left view. The glasses can be passive or active. Passive glasses are used for color (wavelength) or polarization multiplexing of the two views. Anaglyph is the oldest form of 3D display by color multiplexing using red and cyan filters. Polarization multiplexing applies horizontal and vertical (linear), or clockwise and counterclockwise (circular) polarization to the left and right views, respectively. Glasses apply matching polarization to the right and left eyes. The display shows both left and right views laid over each other with polarization matching that of the glasses in every frame. This will lead to some loss of

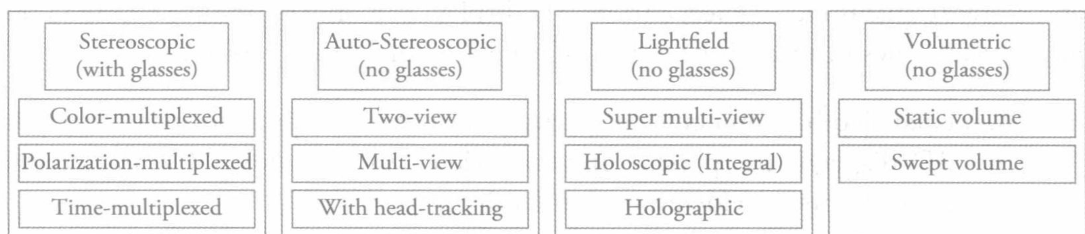



**Figure 2.8** Classification of 3D-display technologies.

spatial resolution since half of the sub-pixels in the display panel will be allocated to the left and right views, respectively, using polarized filters. Active glasses (also called active shutter) present the left image to only the left eye by blocking the view of the right eye while the left image is being displayed and vice versa. The display alternates full-resolution left and right images in sequential order. The active 3D system must assure proper synchronism between the display and glasses. 3D viewing with passive or active glasses is the most developed and commercially available form of 3D display technology. We note that two-view displays lack head-motion parallax and can only provide 3D viewing from a single point of view (from the point where the right and left views have actually been captured) no matter from which angle the viewer looks at the screen. Furthermore, polarization may cause loss of some light due to polarization filter absorption, which may affect scene brightness.

Auto-stereoscopic displays do not require glasses. They can display two views or multiple views. Separation of views can be achieved by different optics technologies, such as parallax barriers or lenticular sheets, so that only certain rays are emitted in certain directions. They can provide head-motion parallax, in addition to binocular depth cues, by either using head-tracking to display two views generated according to head/eye position of the viewer or displaying multiple fixed views. In the former, the need for head-tracking, real-time view generation, and dynamic optics to steer two views in the direction of the viewer gaze increases hardware complexity. In the latter, continuous-motion parallax is not possible with a limited number of views, and proper 3D vision is only possible from some select viewing positions, called sweet spots. In order to determine the number of views, we divide the head-motion range into 2 cm intervals (zones) and present a view for each zone. Then, images seen by the left and right eyes (separated by 6 cm) will be separated by three views. If we allow 4-5 cm head movement toward the left and right, then the viewing range can be covered by a total of eight or nine views. The major drawbacks of autostereoscopic multi-view displays are: i) multiple views are displayed over the same physical screen, sharing sub-pixels between views in a predetermined pattern, which results in loss of spatial resolution; ii) cross-talk between multiple views is unavoidable due to limitations of optics; and iii) there may be noticeable parallax jumps from view to view with a limited number of viewing zones. Due to these reasons, auto-stereoscopic displays have not entered the mass consumer market yet.

State-of-the art stereoscopic and auto-stereoscopic displays have been reviewed in [Ure 11]. Detailed analysis of stereoscopic and auto-stereoscopic displays from a signal-processing perspective and their quality profiles are provided in [Boe 13].

### *Light-Field and Holographic Displays*

Super multi-view (SMV) displays can display up to hundreds of views of a scene taken from different angles (instead of just a right and left view) to create a see-around effect as the viewer slightly changes his/her viewing (gaze) angle. SMV displays employ more advanced optical technologies than just allocating certain sub-pixels to certain views [Ure 11]. The characteristic parameters of a light-field display are spatial, angular, and perceived depth resolution. If the number of views is sufficiently large such that viewing zones are less than 3 mm, two or more views can be displayed within each eye pupil to overcome the accommodation-vergence conflict and offer a real 3D viewing experience. Quality measures for 3D light-field displays have been studied in [Kov 14].

Holographic imaging requires capturing amplitude (intensity), phase differences (interference pattern), and wavelength (color) of a light field using a coherent light source (laser). Holoscopic imaging (or integral imaging) does not require a coherent light source, but employs an array of microlenses to capture and reproduce a 4D light field, where each lens shows a different view depending on the viewing angle.

### *Volumetric Displays*

Different volumetric display technologies aim at creating a 3D viewing experience by means of rendering illumination within a volume that is visible to the unaided eye either directly from the source or via an intermediate surface such as a mirror or glass, which can undergo motion such as oscillation or rotation. They can be broadly classified as swept-volume displays and static volume displays. Swept-volume 3D displays rely on the persistence of human vision to fuse a series of slices of a 3D object, which can be rectangular, disc-shaped, or helical cross-sectioned, into a single 3D image. Static-volume 3D displays partition a finite volume into addressable volume elements, called voxels, made out of active elements that are transparent in "off" state but are either opaque or luminous in "on" state. The resolution of a volumetric display is determined by the number of voxels. It is possible to display scenes with viewing-position-dependent effects (e.g., occlusion) by including transparency (alpha) values for voxels. However, in this case, the scene may look distorted if viewed from positions other than those it was generated for.

The light-field, volumetric, and holographic display technologies are still being developed in major research laboratories around the world and cannot be considered as mature technologies at the time of writing. Note that light-field and volumetric-video representations require orders of magnitude more data (and transmission bandwidth) compared to stereoscopic video. In the following, we cover representations for two-view, multi-view, and super multi-view video.

## 2.3.2 Stereoscopic Video

Stereoscopic two-view video formats can be classified as frame-compatible and full-resolution formats.

Frame-compatible stereo-video formats have been developed to provide 3DTV services over existing digital TV broadcast infrastructures. They employ pixel sub-sampling in order to keep the frame size and rate the same as that of monocular 2D video. Common sub-sampling patterns include side-by-side, top-and-bottom, line interleaved, and checkerboard. Side-by-side format, shown in Figure 2.9(a), applies horizontal subsampling to the left and right views, reducing horizontal resolution by 50%. The subsampled frames are then put together side-by-side. Likewise, top-and-bottom format, shown in Figure 2.9(b), vertically subsamples the left and right views, and stitches them over-under. In the line-interleaved format, the left and right views are again sub-sampled vertically, but put together in an interleaved fashion. Checkerboard format sub-samples left and right views in an offset grid pattern and multiplexes them into a single frame in a checkerboard layout. Among these formats, side-by-side and top-and-bottom are selected as mandatory for broadcast by the latest HDMI specification 1.4a [HDM 13]. Frame-compatible formats are also supported by the stereo and multi-view extensions of the most recent joint MPEG and ITU video-compression standards such as AVC and HEVC (see Chapter 8).

The two-view full resolution stereo is the format of choice for movie and game content. Frame packing, which is a supported format in the HDMI specification version 1.4a, stores frames of left and right views sequentially, without any change in resolution. This full HD stereo-video format requires, in the worst case, twice as much bandwidth as that of monocular video. The extra bandwidth requirement may be kept around 50% by using the Multi-View Video Coding (MVC) standard, which is selected by the Blu-ray Disc Association as the coding format for 3D video.

## 2.3.3 Multi-View Video

Multi-view and super multi-view displays employ multi-view video representations with varying number of views. Since the required data rate increases linearly with the number of views, depth-based representations are more efficient for

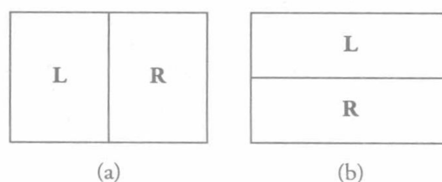



**Figure 2.9** Frame compatible formats: (a) side-by-side; (b) top-bottom.

multi-view video with more than a few views. Depth-based representations also enable: i) generation of desired intermediate views that are not present among the original views by using depth-image based rendering (DIBR) techniques, and ii) easy manipulation of depth effects to adjust vergence vs. accommodation conflict for best viewing comfort.

View-plus-depth has initially been proposed as a stereo-video format, where a single view and associated depth map are transmitted to render a stereo pair at the decoder. It is backward compatible with legacy video using a layered bit stream with an encoded view and encoded depth map as a supplementary layer. MPEG specified a container format for view-plus-depth data, called MPEG-C Part 3 [MPG 07], which was later extended to multi-view-video-plus-depth (MVD) format [Smo 11], where $N$ views and $N$ depth maps are encoded and transmitted to generate $M$ views at the decoder, with $N \leq M$. Te MVD format is illustrated in Figure 2.10, where only 6 views and 6 depth maps per frame are encoded to reconstruct 45 views per frame at the decoder side by using DIBR techniques.

The depth information needs to be accurately captured/computed, encoded, and transmitted in order to render intermediate views accurately using the received reference view and depth map. Each frame of the depth map conveys the distance of the corresponding video pixel from the camera. Scaled depth values, represented by 8 bits, can be regarded as a separate gray-scale video, which can be compressed very efficiently using state-of-the-art video codecs. Depth map typically requires 15–20%

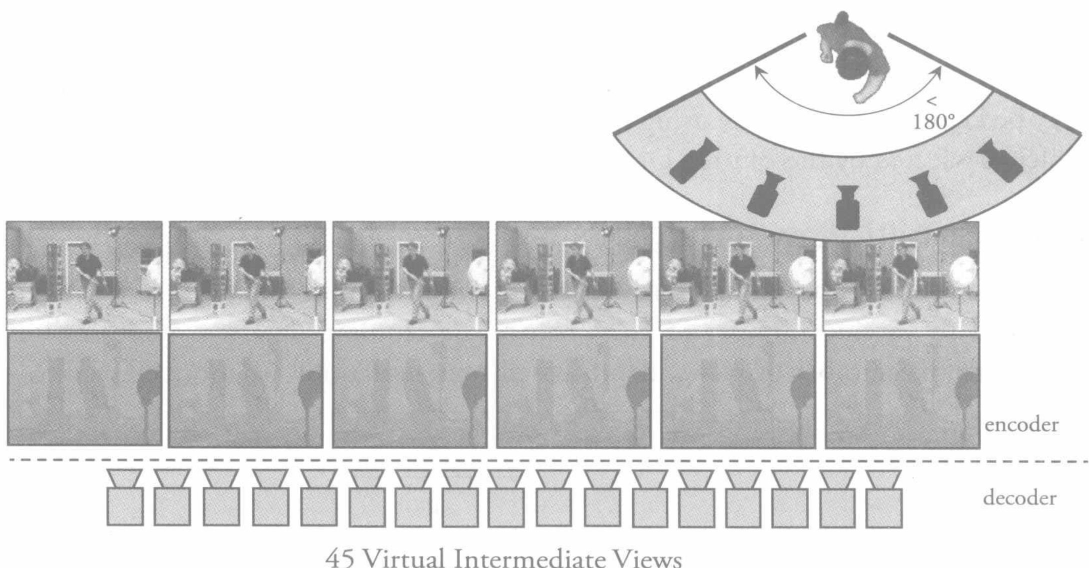



**Figure 2.10** N-view + N depth-map format (courtesy of Aljoscha Smolic).

of the bitrate necessary to encode the original video due to its smooth and less-structured nature.

A difficulty with the view-plus-depth format is generation of accurate depth maps. Although there are time-of-flight cameras that can generate depth or disparity maps, they typically offer limited performance in outdoors environments. Algorithms for depth and disparity estimation by image rectification and disparity matching have been studied in the literature [Kau 07]. Another difficulty is the appearance of regions in the rendered views, which are occluded in the available views. These *disocclusion* regions may be concealed by smoothing the original depth-map data to avoid appearance of holes. Also, it is possible to use multiple view-plus-depth data to prevent disocclusions [Mul 11]. An extension of the view-plus-depth, which allows better modeling of occlusions, is the layered depth video (LDV). LDV provides multiple depth values for each pixel in a video frame.

While high-definition digital-video products have gained universal user acceptance, there are a number of challenges to overcome in bringing 3D video to consumers. Most importantly, advances in autostereoscopic (without glasses) multi-view display technology will be critical for practical usability and consumer acceptance of 3D viewing technology. Availability of high-quality 3D content at home is another critical factor. In summary, both content creators and display manufacturers need further effort to provide consumers with a high-quality 3D experience without viewing discomfort or fatigue and high transition costs. It seems that the TV/consumer electronics industry has moved its focus to bringing ultra-high-definition products to consumers until there is more progress with these challenges.

## 2.4 Digital-Video Applications

Main consumer applications for digital video include digital TV broadcasts, digital cinema, video playback from DVD or blu-ray players, as well as video streaming and videoconferencing over the Internet (wired or wireless) [Pit 13].

### 2.4.1 Digital TV

A digital TV (DTV) broadcasting system consists of video/audio compression, multiplex and transport protocols, channel coding, and modulation subsystems. The biggest single innovation that enabled digital TV services has been advances in video compression since the 1990s. Video-compression standards and algorithms are covered in detail in Chapter 8. Video and audio are compressed separately by different encoders to produce video and audio *packetized elementary streams* (PES). Video and

audio PES and related data are multiplexed into an MPEG *program stream* (PS). Next, one or more PSs are multiplexed into an MPEG *transport stream* (TS). TS packets are 188-bytes long and are designed with synchronization and recovery in mind for transmission in lossy environments. The TS is then modulated into a signal for transmission. Several different modulation methods exist that are specific to the medium of transmission, which are terrestial (fixed reception), cable, satellite, and mobile reception.

There are different digital TV broadcasting standards that are deployed globally. Although they all use MPEG-2 or MPEG-4 AVC/H.264 video compression, more or less similar audio coding, and the same transport stream protocol, their channel coding, transmission bandwidth and modulation systems differ slightly. These include the Advanced Television System Committee (ATSC) in the USA, Digital Video Broadcasting (DVB) in Europe, Integrated Multimedia Broadcasting (ISDB) in Japan, and Digital Terrestial Multimedia Broadcasting in China.

**ATSC Standards**

The first DTV standard was ATSC Standard A/53, which was published in 1995 and was adopted by the Federal Communications Commission in the United States in 1996. This standard supported MPEG-2 Main profile video encoding and 5.1-channel surround sound using Dolby Digital AC-3 encoding, which was standardized as A/52. Support for AVC/H.264 video encoding was added with the ATSC Standard A/72 that was approved in 2008. ATSC signals are designed to use the same 6 MHz bandwidth analog NTSC television channels. Once the digital video and audio signals have been compressed and multiplexed, ATSC uses a 188-byte MPEG transport stream to encapsulate and carry several video and audio programs and metadata. The transport stream is modulated differently depending on the method of transmission:

- Terrestrial broadcasters use 8-VSB modulation that can transmit at a maximum rate of 19.39 Mbit/s. ATSC 8-VSB transmission system adds 20 bytes of Reed-Solomon forward-error correction to create packets that are 208 bytes long.
- Cable television stations operate at a higher signal-to-noise ratio than terrestial broadcasters and can use either 16-VSB (defined by ATSC) or 256-QAM (defined by Society of Cable Telecommunication Engineers) modulation to achieve a throughput of 38.78 Mbit/s, using the same 6-MHz channel.
- There is also an ATSC standard for satellite transmission; however, direct-broadcast satellite systems in the United States and Canada have long used

either DVB-S (in standard or modified form) or a proprietary system such as DSS (Hughes) or DigiCipher 2 (Motorola).

The receiver must demodulate and apply error correction to the signal. Then, the transport stream may be de-multiplexed into its constituent streams before audio and video decoding.

The newest edition of the standard is ATSC-3.0, which employs the HEVC/H.265 video codec, with OFDM instead of 8-VSB for terrestial modulation, allowing for 28 Mbps or more of bandwidth on a single 6-MHz channel.

### DVB Standards

DVB is a suite of standards, adopted by the European Telecommunications Standards Institute (ETSI) and supported by European Broadcasting Union (EBU), which defines the physical layer and data-link layer of the distribution system. The DVB texts are available on the ETSI website. They are specific for each medium of transmission, which we briefly review.

#### *DVB-T and DVB-T2*

DVB-T is the DVB standard for terrestrial broadcast of digital television and was first published in 1997. It specifies transmission of MPEG transport streams, containing MPEG-2 or H.264/MPEG-4 AVC compressed video, MPEG-2 or Dolby Digital AC-3 audio, and related data, using coded orthogonal frequency-division multiplexing (COFDM) or OFDM modulation. Rather than carrying data on a single radio frequency (RF) channel, COFDM splits the digital data stream into a large number of lower rate streams, each of which digitally modulates a set of closely spaced adjacent sub-carrier frequencies. There are two modes: 2K-mode (1,705 sub-carriers that are 4 kHz apart) and 8K-mode (6,817 sub-carriers that are 1 kHz apart). DVB-T offers three different modulation schemes (QPSK, 16QAM, 64QAM). It was intended for DTV broadcasting using mainly VHF 7 MHz and UHF 8 MHz channels. The first DVB-T broadcast was realized in the UK in 1998. The DVB-T2 is the extension of DVB-T that was published in June 2008. With several technical improvements, it provides a minimum 30% increase in payload, under similar channel conditions compared to DVB-T. The ETSI adopted the DVB-T2 in September 2009.

#### *DVB-S and DVB-S2*

DVB-S is the original DVB standard for satellite television. Its first release dates back to 1995, while development lasted until 1997. The standard only specifies physical

link characteristics and framing for delivery of MPEG transport stream (MPEG-TS) containing MPEG-2 compressed video, MPEG-2 or Dolby Digital AC-3 audio, and related data. The first commercial application was in Australia, enabling digitally broadcast, satellite-delivered television to the public. DVB-S has been used in both multiple-channel per carrier and single-channel per carrier modes for broadcast network feeds and direct broadcast satellite services in every continent of the world, including Europe, the United States, and Canada.

DVB-S2 is the successor of the DVB-S standard. It was developed in 2003 and ratified by the ETSI in March 2005. DVB-S2 supports broadcast services including standard and HDTV, interactive services including Internet access, and professional data content distribution. The development of DVB-S2 coincided with the introduction of HDTV and H.264 (MPEG-4 AVC) video codecs. Two new key features that were added compared to the DVB-S standard are:

- A powerful coding scheme, Irregular Repeat-Accumulate codes, based on a modern LDPC code, with a special structure for low encoding complexity.
- Variable coding and modulation (VCM) and adaptive coding and modulation (ACM) modes to optimize bandwidth utilization by dynamically changing transmission parameters.

Other features include enhanced modulation schemes up to 32-APSK, additional code rates, and introduction of a generic transport mechanism for IP packet data including MPEG-4 AVC video and audio streams, while supporting backward compatibility with existing DVB-S transmission. The measured DVB-S2 performance gain over DVB-S is around a 30% increase of available bitrate at the same satellite transponder bandwidth and emitted signal power. With improvements in video compression, an MPEG-4 AVC HDTV service can now be delivered in the same bandwidth used for an early DVB-S based MPEG-2 SDTV service. In March 2014, the DVB-S2X specification was published as an optional extension adding further improvements.

### *DVB-C and DVB-C2*

The DVB-C standard is for broadcast transmission of digital television over cable. This system transmits an MPEG-2 or MPEG-4 family of digital audio/digital video stream using QAM modulation with channel coding. The standard was first published by the ETSI in 1994, and became the most widely used transmission system for digital cable television in Europe. It is deployed worldwide in systems ranging

from larger cable television networks (CATV) to smaller satellite master antenna TV (SMATV) systems.

The second-generation DVB cable transmission system DVB-C2 specification was approved in April 2009. DVB-C2 allows bitrates up to 83.1 Mbit/s on an 8 MHz channel when using 4096-QAM modulation, and up to 97 Mbit/s and 110.8 Mbit/s per channel when using 16384-QAM and 65536-AQAM modulation, respectively. By using state-of-the-art coding and modulation techniques, DVB-C2 offers more than a 30% higher spectrum efficiency under the same conditions, and the gains in downstream channel capacity are greater than 60% for optimized HFC networks. These results show that the performance of the DVB-C2 system gets so close to the theoretical Shannon limit that any further improvements would most likely not be able to justify the introduction of a disruptive third generation cable-transmission system.

There is also a DVB-H standard for terrestrial mobile TV broadcasting to hand-held devices. The competitors of this technology have been the 3G cellular-system-based MBMS mobile-TV standard, the ATSC-M/H format in the United States, and the Qualcomm MediaFLO. DVB-SH (satellite to handhelds) and DVB-NGH (Next Generation Handheld) are possible future enhancements to DVB-H. However, none of these technologies have been commercially successful.

### 2.4.2 Digital Cinema

Digital cinema refers to digital distribution and projection of motion pictures as opposed to use of motion picture film. A digital cinema theatre requires a digital projector (instead of a conventional film projector) and a special computer server. Movies are supplied to theatres as digital files, called a Digital Cinema Package (DCP), whose size is between 90 gigabytes (GB) and 300 GB for a typical feature movie. The DCP may be physically delivered on a hard drive or can be downloaded via satellite. The encrypted DCP file first needs to be copied onto the server. The decryption keys, which expire at the end of the agreed upon screening period, are supplied separately by the distributor. The keys are locked to the server and projector that will screen the film; hence, a new set of keys are required to show the movie on another screen. The playback of the content is controlled by the server using a playlist.

#### Technology and Standards

Digital cinema projection was first demonstrated in the United States in October 1998 using Texas Instruments' DLP projection technology. In January 2000, the

Society of Motion Picture and Television Engineers, in North America, initiated a group to develop digital cinema standards. The Digital Cinema Initiative (DCI), a joint venture of six major studios, was established in March 2002 to develop a system specification for digital cinema to provide robust intellectual property protection for content providers. DCI published the first version of a specification for digital cinema in July 2005. Any DCI-compliant content can play on any DCI-compliant hardware anywhere in the world.

Digital cinema uses high-definition video standards, aspect ratios, or frame rates that are slightly different than HDTV and UHDTV. The DCI specification supports 2K (2048 × 1080 or 2.2 Mpixels) at 24 or 48 frames/sec and 4K (4096 × 2160 or 8.8 Mpixels) at 24 frames/sec modes, where resolutions are represented by the horizontal pixel count. The 48 frames/sec is called high frame rate (HFR). The specification employs the ISO/IEC 15444-1 JPEG2000 standard for picture encoding, and the CIE XYZ color space is used at 12 bits per component encoded with a 2.6 gamma applied at projection. It ensures that 2K content can play on 4K projectors and vice versa.

**Digital Cinema Projectors**

Digital cinema projectors are similar in principle to other digital projectors used in the industry. However, they must be approved by the DCI for compliance with the DCI specifications: i) they must conform to the strict performance requirements, and ii) they must incorporate anti-piracy protection to protect copyrights. Major DCI-approved digital cinema projector manufacturers include Christie, Barco, NEC, and Sony. The first three manufactuers have licensed the DLP technology from Texas Instruments, and Sony uses its own SXRD technology. DLP projectors were initially available in 2K mode only. DLP projectors became available in both 2K and 4K in early 2012, when Texas Instruments' 4K DLP chip was launched. Sony SXRD projectors are only manufactured in 4K mode.

DLP technology is based on digital micromirror devices (DMDs), which are chips whose surface is covered by a large number of microscopic mirrors, one for each pixel; hence, a 2K chip has about 2.2 million mirrors and a 4K chip about 8.8 million. Each mirror vibrates several thousand times a second between on and off positions. The proportion of the time the mirror is in each position varies according to the brightness of each pixel. Three DMD devices are used for color projection, one for each of the primary colors. Light from a Xenon lamp, with power between 1 kW and 7 kW, is split by color filters into red, green, and blue beams that are directed at the appropriate DMD.

Transition to digital projection in cinemas is ongoing worldwide. According to the National Association of Theatre Owners, 37,711 screens out of 40,048 in the United States had been converted to digital and about 15,000 were 3D capable as of May 2014.

### *3D Digital Cinema*

The number of 3D-capable digital cinema theatres is increasing with wide interest of audiences in 3D movies and an increasing number of 3D productions. A 3D-capable digital cinema video projector projects right-eye and left-eye frames sequentially. The source video is produced at 24 frames/sec per eye; hence, a total of 48 frames/sec for right and left eyes. Each frame is projected three times to reduce flicker, called triple flash, for a total of 144 times per second. A silver screen is used to maintain light polarization upon reflection. There are two types of stereoscopic 3D viewing technology where each eye sees only its designated frame: i) glasses with polarizing filters oriented to match projector filters, and ii) glasses with liquid crystal (LCD) shutters that block or transmit light in sync with the projectors. These technologies are provided under the brands RealD, MasterImage, Dolby 3D, and XpanD.

The polarization technology combines a single 144-Hz digital projector with either a polarizing filter (for use with polarized glasses and silver screens) or a filter wheel. *RealD* 3D cinema technology places a push-pull electro-optical liquid crystal modulator called a ZScreen in front of the projector lens to alternately polarize each frame. It circularly polarizes frames clockwise for the right eye and counter-clockwise for the left eye. *MasterImage* uses a filter wheel that changes the polarity of the projector's light output several times per second to alternate the left-and-right-eye views. *Dolby* 3D also uses a filter wheel. The wheel changes the wavelengths of colors being displayed, and tinted glasses filter these changes so the incorrect wavelength cannot enter the wrong eye. The advantage of circular polarization over linear polarization is that viewers are able to slightly tilt their head without seeing double or darkened images.

The *XpanD* system alternately flashes the images for each eye that viewers observe using electronically synchronized glasses The viewer wears electronic glasses whose LCD lenses alternate between clear and opaque to show only the correct image at the correct time for each eye. XpanD uses an external emitter that broadcasts an invisible infrared signal in the auditorium that is picked up by glasses to synchronize the shutter effect.

*IMAX Digital 3D* uses two separate 2K projectors that represent the left and right eyes. They are separated by a distance of 64 mm (2.5 in), which is the average distance

between a human's eyes. The two 2K images are projected over each other (superposed) on a silver screen with proper polarization, which makes the image brighter. Right and left frames on the screen are directed only to the correct eye by means of polarized glasses that enable the viewer to see in 3D. Note that IMAX theatres use the original 15/70 IMAX higher resolution frame format on larger screens.

### 2.4.3 Video Streaming over the Internet

Video streaming refers to delivery of media over the Internet, where the client player can begin playback before the entire file has been sent by the server. A server-client streaming system consists of a streaming server and a client that communicate using a set of standard protocols. The client may be a standalone player or a plugin as part of a Web browser. The streaming session can be a video-on-demand request (sometimes called a pull-application) or live Internet broadcasting (called a push-application). In a video-on-demand session, the server streams from a pre-encoded and stored file. Live streaming refers to live content delivered in real-time over the Internet, which requires a live camera and a real-time encoder on the server side.

Since the Internet is a best-effort channel, packets may be delayed or dropped by the routers and the effective end-to-end bitrates fluctuate in time. Adaptive streaming technologies aim to adapt the video-source (encoding) rate according to an estimate of the available end-to-end network rate. One possible way to do this is stream switching, where the server encodes source video at multiple pre-selected bitrates and the client requests switching to the stream encoded at the rate that is closest to its network access rate. A less commonly deployed solution is based on scalable video coding, where one or more enhancement layers of video may be dropped to reduce the bitrate as needed.

In the server-client model, the server sends a different stream to each client. This model is not scalable, since server load increases linearly with the number of stream requests. Two solutions to solve this problem are multicasting and peer-to-peer (P2P) streaming. We discuss the server-client, multicast, and P2P streaming models in more detail below.

#### Server-Client Streaming

This is the most commonly used streaming model on the Internet today. All video streaming systems deliver video and audio streams by using a streaming protocol built on top of transmission control protocol (TCP) or user datagram protocol (UDP). Streaming solutions may be based on open-standard protocols published by

the Internet Engineering Task Force (IETF) such as RTP/UDP or HTTP/TCP, or may be proprietary systems, where RTP stands for real-time transport protocol and HTTP stands for hyper-text transfer protocol.

### *Streaming Protocols*

Two popular streaming protocols are Real-Time Streaming Protocol (RTSP), an open standard developed and published by the IETF as RFC 2326 in 1998, and Real Time Messaging Protocol (RTMP), a proprietary solution developed by Adobe Systems.

RTSP servers use the Real-time Transport Protocol (RTP) for media stream delivery, which supports a range of media formats (such as AVC/H.264, MJPEG, etc.). Client applications include QuickTime, Skype, and Windows Media Player. Android smartphone platforms also include support for RTSP as part of the 3GPP standard.

RTMP is primarily used to stream audio and video to Adobe's Flash Player client. The majority of streaming videos on the Internet is currently delivered via RTMP or one of its variants due to the success of the Flash Player. RTMP has been released for public use. Adobe has included support for adaptive streaming into the RTMP protocol.

The main problem with UDP-based streaming is that streams are frequently blocked by firewalls, since they are not being sent over HTTP (port 80). In order to circumvent this problem, protocols have been extended to allow for a stream to be encapsulated within HTTP requests, which is called tunneling. However, tunneling comes at a performance cost and is often only deployed as a fallback solution. Streaming protocols also have secure variants that use encryption to protect the stream.

### *HTTP Streaming*

Streaming over HTTP, which is a more recent technology, works by breaking a stream into a sequence of small HTTP-based file downloads, where each download loads one short *chunk* of the whole stream. All flavors of HTTP streaming include support for adaptive streaming (bitrate switching), which allows clients to dynamically switch between different streams of varying quality and chunk size during playback, in order to adapt to changing network conditions and available CPU resources. By using HTTP, firewall issues are generally avoided. Another advantage of HTTP streaming is that it allows HTTP chunks to be cached within ISPs or

corporations, which would reduce the bandwidth required to deliver HTTP streams, in contrast to video streamed via RTMP.

Different vendors have implemented different HTTP-based streaming solutions, which all use similar mechanisms but are incompatible; hence, they all require the vendor's own software:

- HTTP Live Streaming (HLS) by Apple is an HTTP-based media streaming protocol that can dynamically adjust movie playback quality to match the available speed of wired or wireless networks. HTTP Live Streaming can deliver streaming media to an iOS app or HTML5-based website. It is available as an IETF Draft (as of October 2014) [Pan 14].
- Smooth Streaming by Microsoft enables adaptive streaming of media to clients over HTTP. The format specification is based on the ISO base media file format. Microsoft provides Smooth Streaming Client software development kits for Silverlight and Windows Phone 7.
- HTTP Dynamic Streaming (HDS) by Adobe provides HTTP-based adaptive streaming of high-quality AVC/H.264 or VP6 video for a Flash Player client platform.

MPEG-DASH is the first adaptive bit-rate HTTP-based streaming solution that is an international standard, published in April 2012. MPEG-DASH is audio/video codec agnostic. It allows devices such as Internet-connected televisions, TV set-top boxes, desktop computers, smartphones, tablets, etc., to consume multimedia delivered via the Internet using previously existing HTTP web server infrastructure, with the help of adaptive streaming technology. Standardizing an adaptive streaming solution aims to provide confidence that the solution can be adopted for universal deployment, compared to similar proprietary solutions such as HLS by Apple, Smooth Streaming by Microsoft, or HDS by Adobe. An implementation of MPEG-DASH using a content centric networking (CCN) naming scheme to identify content segments is publicly available [Led 13]. Several issues still need to be resolved, including legal patent claims, before DASH can become a widely used standard.

#### Multicast and Peer-to-Peer (P2P) Streaming

Multicast is a one-to-many delivery system, where the source server sends each packet only once, and the nodes in the network replicate packets only when necessary to reach multiple clients. The client nodes send join and leave messages, e.g., as in the

case of Internet television when the user changes the TV channel. In P2P streaming, clients (peers) forward packets to other peers (as opposed to network nodes) to minimize the load on the source server.

The multicast concept can be implemented at the IP or application level. The most common transport layer protocol to use multicast addressing is the User Datagram Protocol (UDP). IP multicast is implemented at the IP routing level, where routers create optimal distribution paths for datagrams sent to a multicast destination address. IP multicast has been deployed in enterprise networks and multimedia content delivery networks, e.g., in IPTV applications. However, IP multicast is not implemented in commercial Internet backbones mainly due to economic reasons. Instead, application layer multicast-over-unicast overlay services for application-level group communication are widely used.

In media streaming over P2P overlay networks, each peer forwards packets to other peers in a live media streaming session to minimize the load on the server. Several protocols that help peers find a relay peer for a specified stream exist [Gu 14]. There are P2PTV networks based on real-time versions of the popular file-sharing protocol BitTorrent. Some P2P technologies employ the multicast concept when distributing content to multiple recipients, which is known as peercasting.

### 2.4.4 Computer Vision and Scene/Activity Understanding

Computer vision is a discipline of computer science that aims to duplicate abilities of human vision by processing and understanding digital images and video. It is such a large field that it is the subject of many excellent textbooks [Har 04, For 11, Sze 11]. The visual data to be processed can be still images, video sequences, or views from multiple cameras. Computer vision is generally divided into high-level and low-level vision. High-level vision is often considered as part of artificial intelligence and is concerned with the theory of learning and pattern recognition with application to object/activity recognition in order to extract information from images and video. We mention computer vision here because many of the problems addressed in image/video processing and low-level vision are common. Low-level vision includes many image- and video-processing tasks that are the subject of this book such as edge detection, image enhancement and restoration, motion estimation, 3D scene reconstruction, image segmentation, and video tracking. These low-level vision tasks have been used in many computer-vision applications, including road monitoring, military surveillance, and robot navigation. Indeed, several of the methods discussed in this book have been developed by computer-vision researchers.

## 2.5 Image and Video Quality

Video quality may be measured by the quality of experience of viewers, which can usually be reliably measured by subjective methods. There have been many studies to develop objective measures of video quality that correlate well with subjective evaluation results [Cho 14, Bov 13]. However, this is still an active research area. Since analog video is becoming obsolete, we start by defining some visual artifacts related to digital video that are the main cause of loss of quality of experience.

### 2.5.1 Visual Artifacts

Artifacts are visible distortions in images/videos. We can classify visual artifacts as spatial and temporal artifacts. Spatial artifacts, such as blur, noise, ringing, and blocking, are most disturbing in still images but may also be visible in video. In addition, in video, temporal freeze and skipped frames are important causes of visual disturbance and, hence, loss of quality of experience.

Blur refers to lack or loss of image sharpness (high spatial frequencies). The main causes of blur are insufficient spatial resolution, defocus, and/or motion between camera and the subject. According to the Nyquist sampling theorem, the highest horizontal and vertical spatial frequencies that can be represented is determined by the sampling rate (pixels/cm), which relates to image resolution. Consequently, low-resolution images cannot contain high spatial frequencies and appear blurred. Defocus blur is due to incorrect focus of the camera, which may be due to depth of field. Motion blur is caused by relative movement of the subject and camera while the shutter is open. It may be more noticeable in imaging darker scenes since the shutter has to remain open for longer time.

Image noise refers to low amplitude, high-frequency random fluctuations in the pixel values of recorded images. It is an undesirable by-product of image capture, which can be produced by film grain, photo-electric sensors, and digital camera circuitry, or image compression. It is measured by signal-to-noise ratio. Noise due to electronic fluctuations can be modeled by a white, Gaussian random field, while noise due to LCD sensor imperfections is usually modeled as impulsive (salt-and-pepper) noise. Noise at low-light (signal) levels can be modeled as speckle noise.

Image/video compression also generates noise, known as quantization noise and mosquito noise. Quantization or truncation of the DCT/wavelet transform coefficients results in quantization noise. Mosquito noise is temporal noise, i.e., flickering-like luminance/chrominance fluctuations as a consequence of differences in coding observed in smoothly textured regions or around high contrast edges in consecutive frames of video.

Ringing and blocking artifacts, which are by-products of DCT image/video compression, are also observed in compressed images/video. Ringing refers to oscillations around sharp edges. It is caused by sudden truncation of DCT coefficients due to coarse quantization (also known as the Gibbs effect). DCT is usually taken over 8 × 8 blocks. Coarse quantization of DC coefficients may cause mismatch of image mean over 8 × 8 blocks, which results in visible block boundaries known as blocking artifacts.

Skip frame and freeze frame are the result of video transmission over unreliable channels. They are caused by video packets that are not delivered on time. When video packets are late, there are two options: skip late packets and continue with the next packet, which is delivered on time, or wait (freeze) until the late packets arrive. Skipped frames result in motion jerkiness and discontinuity, while freeze frame refers to complete stopping of action until the video is rebuffered.

Visibility of artifacts is affected by the viewing conditions, as well as the type of image/video content as a result of spatial and temporal-masking effects. For example, spatial-image artifacts that are not visible in full-motion video may be higly objectionable when we freeze frame.

### 2.5.2 Subjective Quality Assessment

Measurement of subjective video quality can be challenging because many parameters of set-up and viewing conditions, such as room illumination, display type, brightness, contrast, resolution, viewing distance, and the age and educational level of experts, can influence the results. The selection of video content and the duration also affect the results. A typical subjective video quality evaluation procedure consists of the following steps:

1. Choose video sequences for testing
2. Choose the test set-up and settings of system to evaluate
3. Choose a test method (how sequences are presented to experts and how their opinion is collected: DSIS, DSCQS, SSCQE, DSCS)
4. Invite sufficient number and types of experts (18 or more is recommended)
5. Carry out testing and calculate the mean expert opinion scores (MOS) for each test set-up

In order to establish meaningful subjective assessment results, some test methods, grading scales, and viewing conditions have been standardized by ITU-T Recommendation BT.500-11 (2002) "Methodology for the subjective assessment of the quality of television pictures." Some of these test methods are double stimulus where

viewers rate the quality or change in quality between two video streams (reference and impaired). Others are single stimulus where viewers rate the quality of just one video stream (the impaired). Examples of the former are the double stimulus impairment scale (DSIS), double stimulus continuous quality scale (DSCQS), and double stimulus comparison scale (DSCS) methods. An example of the latter is the single stimulus continuous quality evaluation (SSCQE) method. In the DSIS method, observers are first presented with an unimpaired reference video, then the same video impaired, and he/she is asked to vote on the second video using an impairment scale (from "impairments are imperceptible" to "impairments are very annoying"). In the DSCQS method, the sequences are again presented in pairs: the reference and impaired. However, observers are not told which one is the reference and are asked to assess the quality of both. In the series of tests, the position of the reference is changed randomly. Different test methodologies have claimed advantages for different cases.

### 2.5.3 Objective Quality Assessment

The goal of *objective* image quality assessment is to develop quantitative measures that can automatically predict perceived image quality [Bov 13]. Objective image/video quality metrics are mathematical models or equations whose results are expected to correlate well with subjective assessments. The goodness of an objective video-quality metric can be assessed by computing the correlation between the objective scores and the subjective test results. The most frequently used correlation coefficients are the Pearson linear correlation coefficient, Spearman rank-order correlation coefficient, kurtosis, and the outliers ratio.

Objective metrics are classified as full reference (FR), reduced reference (RR), and no-reference (NR) metrics, based on availability of the original (high-quality) video, which is called the reference. FR metrics compute a function of the difference between every pixel in each frame of the test video and its corresponding pixel in the reference video. They cannot be used to evaluate the quality of the received video, since a reference video is not available at the receiver end. RR metrics extract some features of both videos and compare them to give a quality score. Only some features of the reference video must be sent along with the compressed video in order to evaluate the received video quality at the receiver end. NR metrics assess the quality of a test video without any reference to the original video.

#### Objective Image/Video Quality Measures

Perhaps the most well-established methodology for FR objective image and video quality evaluation is pixel-by-pixel comparison of image/video with the reference.

The peak signal-to-noise ratio (PSNR) measures the logarithm of the ratio of the maximum signal power to the mean square difference (MSE), given by

$$PSNR = 10\log_{10}\left(\frac{255^2}{MSE}\right)$$

where the MSE between the test video $\hat{s}[n_1,n_2,k]$, which is $N_1 \times N_2$ pixels and $N_3$ frames long, and reference video $s[n_1,n_2,k]$ with the same size, can be computed by

$$MSE = \frac{1}{\mathrm{N_1 N_2 N_3}} \sum_{n_1=0}^{N_1}\sum_{n_2=0}^{N_2}\sum_{k=0}^{N_3}(s[n_1,n_2,k]-\hat{s}[n_1,n_2,k])^2$$

Some have claimed that PSNR may not correlate well with the perceived visual quality since it does not take into account many characteristics of the human visual system, such as spatial- and temporal-masking effects. To this effect, many alternative FR metrics have been proposed. They can be classified as those based on *structural similarity* and those based on *human vision models*.

The structural similarity index (SSIM) is a structural image similarity based FR metric that aims to measure perceived change in structural information between two $N \times N$ luminance blocks $\mathbf{x}$ and $\mathbf{y}$, with means $\mu_x$ and $\mu_y$ and variances $\sigma_x^2$ and $\sigma_y^2$, respectively. It is given by [Wan 04]

$$SSIM(\mathbf{x},\mathbf{y}) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{\left(\mu_x^2 + \mu_y^2 + c_1\right)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

where $\sigma_{xy}$ is the covariance between windows $\mathbf{x}$ and $\mathbf{y}$ and $c_1$ and $c_2$ are small constants to avoid division by very small numbers.

Perceptual evaluation of video quality (PEVQ) is a vision-model-based FR metric that analyzes pictures pixel-by-pixel after a temporal alignment (registration) of corresponding frames of reference and test video. PEVQ aims to reflect how human viewers would evaluate video quality based on subjective comparison and outputs mean opinion scores (MOS) in the range from 1 (bad) to 5 (excellent).

VQM is an RR metric that is based on a general model and associated calibration techniques and provides estimates of the overall impressions of subjective video quality [Pin 04]. It combines perceptual effects of video artifacts including blur, noise, blockiness, color distortions, and motion jerkiness into a single metric.

NR metrics can be used for monitoring quality of compressed images/video or video streaming over the Internet. Specific NR metrics have been developed for

quantifying such image artifacts as noise, blockiness, and ringing. However, the ability of these metrics to make accurate quality predictions are usually satisfactory only in a limited scope, such as for JPEG/JPEG2000 images.

The International Telecommunications Union (ITU) Video Quality Experts Group (VQEG) standardized some of these metrics, including the PEVQ, SSIM, and VQM, as ITU-T Rec. J.246 (RR) and J.247 (FR) in 2008 and ITU-T Rec. J.341 (FR HD) in 2011. It is perhaps useful to distinguish the performance of these structural similarity and human vision model based metrics on still images and video. It is fair to say these metrics have so far been more successful on still images than video for objective quality assessment.

### Objective Quality Measures for Stereoscopic 3D Video

FR metrics for evaluation of 3D image/video quality is technically not possible, since the 3D signal is formed only in the brain. Hence, objective measures based on a stereo pair or video-plus-depth-maps should be considered as RR metrics. It is generally agreed upon that 3D quality of experience is related to at least three factors:

- Quality of display technology (cross-talk)
- Quality of content (visual discomfort due to accomodation-vergence conflict)
- Encoding/transmission distortions/ artifacts

In addition to those artifacts discussed in Section 2.5.1, the main factors in 3D video quality of experience are visual discomfort and depth perception. As discussed in Section 2.1.4, visual discomfort is mainly due to the conflict between accommodation and vergence and cross-talk between the left and right views. Human perception of distortions/artifacts in 3D stereo viewing is not fully understood yet. There have been some preliminary works on quantifying visual comfort and depth perception [Uka 08, Sha 13]. An overview of evaluation of stereo and multi-view image/video quality can be found in [Win 13]. There are also some studies evaluating the perceptual quality of symmetrically and asymmetrically encoded stereoscopic videos [Sil 13].

## References

[Boe 13] Boev, A., R. Bregovic, and A. Gotchev, "Signal processing for stereoscopic and multiview 3D displays," chapter in *Handbook of Signal Processing Systems, Second Edition*, (ed. S. Bhattacharyya, E. Deprettere, R. Leupers, and J. Takala), pp. 3–47, New York, NY: Springer, 2013.

[Bov 13] Bovik, A. C., "Automatic prediction of perceptual image and video quality," Proc. of the IEEE, vol. 101, no. 9, pp. 2008–2024, Sept. 2013.

[Bre 07] Breitmeyer, B. G., "Visual masking: past accomplishments, present status, future developments," Adv. Cogn. Psychol 3, 2007.

[Cho 14] Choi, L. K., Y. Liao, A. C. Bovik, "Video QoE models for the compute continuum," IEEE ComSoc MMTC E-Letter, vol. 8, no. 5, pp. 26–29, Sept. 2013.

[Dub 10] Dubois, E., *The Structure and Properties of Color Spaces and the Representation of Color Images,* Morgan & Claypool, 2010.

[Fer 01] Ferwerda, J. A., "Elements of early vision for computer graphics, IEEE Computer Graphics and Application, vol. 21, no. 5, pp. 22–33, Sept./Oct. 2001.

[For 11] Forsyth, David A. and Jean Ponce, *Computer Vision: A Modern Approach, Second Edition*, Upper Saddle River, NJ: Prentice Hall, 2011.

[Gon 07] Gonzalez, Rafael C. and Richard E. Woods, *Digital Image Processing, Third Edition*, Upper Saddle River, NJ: Prentice Hall, 2007.

[Gra 10] Granados, M., B. Ajdin, M. Wand, C. Theobalt, H-P. Seidel, and H. P.A. Lensch, "Optimal HDR reconstruction with linear digital cameras," IEEE Int. Conf. Computer Vision and Pattern Recognition (CVPR), pp. 215–222, June 2010.

[Gu 14] Gu, Y., et al., Survey of P2P Streaming Applications, IETF draft-ietf-ppsp-survey-08, April 2014.

[Har 04] Hartley, R. I. and A. Zisserman, *Multiple View Geometry in Computer Vision, Second Edition*, New York, NY: Cambridge University Press, 2004.

[HDM 13] High definition multimedia interface (HDMI). http://www.hdmi.org/index.aspx

[Hub 88] Hubel, D. H., "Eye, Brain, and Vision, Vol. 22, Scientific American Library," distributed by W. H. Freeman & Co., New York, 1988. http://hubel.med.harvard.edu

[IEC 00] IEC 61966-2-1:2000, Multimedia systems and equipment - Colour measurement and management - Colour management - Default RGB colour space: sRGB, Sept. 2000.

[ITU 02] ITU-R Rec. BT.709, Parameter values for the HDTV standards for production and international program exchange, April 2002. http://www.itu.int/rec/R-REC-BT.709

[ITU 11] ITU-R Rec. BT.601, Studio encoding parameters of digital television for standard 4:3 and wide screen 16:9 aspect ratios, March 2011. http://www.itu.int/rec/R-REC-BT.601/

[ITU 12] ITU-R Rec. BT.2020, Parameter values for ultra-high definition television systems for production and international program exchange, August 2012. http://www.itu.int/rec/R-REC-BT.2020-0-201208-I

[Kau 07] Kauff, P., et al., "Depth map creation and image-based rendering for advanced 3DTV services providing interoperability and scalability," Signal Processing: Image Communication, vol. 22, pp. 217–234, 2007.

[Kov 14] Kovacs, P. T., A. Boev, R. Bregovic, and A. Gotchev, "Quality measurements of 3D light-field displays," Int. Workshop on Video Processing and Quality metrics for Consumer Electronics (VPQM), Chandler, AR, USA, Jan. 30–31, 2014.

[Led 13] Lederer, S., C. Muller, B. Rainer, C. Timmerer, and H. Hellwagner, "An experimental analysis of dynamic adaptive streaming over HTTP in content centric networks", in Proc. of IEEE Int. Conf. on Multimedia and Expo (ICME), San Jose, USA, July 2013.

[Mul 85] Mullen, K. T., "The contrast sensitivity of human colour vision to red-green and blue-yellow chromatic gratings," J. Physiol., 1985.

[Nes 67] van Nes, F. L., J. J. Koenderink, H. Nas, and M. A. Bouman, "Spatiotemporal modulation transfer in the human eye," Jour. of the Optical Society of America, vol. 57, no. 9, Sept. 1967.

[MPG 07] ISO/IEC 23002-3:2007 Information technology - MPEG video technologies - Part 3: Representation of auxiliary video and supplemental information, 2007.

[Mul 11] Muller, K., P. Merkle, and T. Wiegand, "3-D video representation using depth maps," Proc. of the IEEE, vol. 99, no. 4, pp. 643–656, April 2011.

[Pan 14] Pantos, R. and W. May, HTTP Live Streaming, IETF draft-pantos-http-live-streaming-14, October 2014.

[Pin 04] Pinson, M. and S. Wolf, "A new standardized method for objectively measuring video quality," IEEE Trans. on Broadcasting, vol. 50, no.3, pp. 312–322, Sept. 2004.

[Pit 13] Pitas, I., *Digital Video and Television,* Ioannis Pitas: 2013.

[Rei 07] Reinhard, E., T. Kunkel, Y. Marion, J. Brouillat, R. Cozot and K. Bouatouch, "Image display algorithms for high and low dynamic range display devices," Jour. of the Society for Information Display, vol. 15 (12), pp. 997–1014, 2007.

[See 04] Seetzen, H., W. Heidrich, W. Stuerzlinger, G. Ward, L. Whitehead, M. Trentacoste, A. Ghosh, and A. Vorozcovs, "High dynamic range display systems," Proc. ACM SIGGRAPH, 2004.

[Sha 13] Shao, F., W. Lin, S. Gu, G. Jiang, and T. Srikanthan, "Perceptual full-reference quality assessment of stereoscopic images by considering binocular visual characteristics." IEEE Trans. on Image Proc., vol. 22, no. 5, pp. 1940–53, May 2013.

[Sha 98] Sharma, G., M. J. Vrhel, and H. J. Trussell, "Color imaging for multimedia," Proc. of the IEEE, vol. 86, no. 6, June 1998.

[Smo 11] Smolic, A., "3D video and free viewpoint video - From capture to display," Pattern Recognition, vol. 44, no. 9, pp. 1958–1968, Sept. 2011.

[Suc 11] Suchow, J. W. and G. A. Alvarez, "Motion silences awareness of visual change," Curr. Biol., vol. 21, no. 2, pp.140–143, Jan. 2011.

[Sze 11] Szeliski, R., *Computer Vision: Algorithms and Applications*, New York, NY: Springer, 2011.

[Tru 93] Trussell, H. J., "DSP solutions run the gamut for color systems," IEEE Signal Processing Mag., pp. 8–23, Apr. 1993.

[Uka 08] Ukai, K., and P. A. Howarth, "Visual fatigue caused by viewing stereoscopic motion images: Background, theories, and observations," *Displays,* vol. 29, pp. 106–116, Mar. 2008.

[Ure 11] Urey, H., K. V. Chellapan, E. Erden, and P. Surman, "State of the art in stereoscopic and autostereoscopic displays," Proc. of the IEEE, vol. 99, no. 4, pp. 540–555, April 2011.

[Wad 96] Wade, N. J., "Descriptions of visual phenomena from Aristotle to Wheatstone," *Perception,* vol. 25, no. 10, pp. 1137–1175, 1996.

[Wan 95] Wandell, B., *Foundations of Vision,* Sunderland, MA: Sinauer Associates, 1995.

[Wan 04] Wang, Z., A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," IEEE Trans. on Image Processing, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[Win 13] Winkler, S. and D. Min, "Stereo/multiview picture quality: Overview and recent advances," Signal Processing: Image Communication, vol. 28, no. 10, pp. 1358–1373, Nov. 2013.

CHAPTER 3

# Image Filtering

*An in-depth understanding of image-processing methods is essential for a rigorous study of digital-video processing. This chapter introduces essential image-filtering operations, such as Gaussian and bi-lateral filtering, gradient estimation, and image interpolation, that are required for motion estimation, as well as foundations of some ill-posed problems such as denoising, restoration, and super-resolution.*

Image filtering refers to processing of an input image to produce either a better-looking output image by contrast/sharpness and/or signal-to-noise ratio enhancement, or to compute some low-level image features such as edges, corners, or spatial-gradient values that may be used in subsequent image processing. This chapter discusses most common linear and nonlinear image-filtering operations including image smoothing, image re-sampling (decimation and interpolation), gradient estimation, edge detection, image enhancement, image denoising, image deblurring (restoration), and image in-painting.

A common practice in processing of color images is to first convert an RGB image into the luminance-chrominance (YCrCb) domain, process the luminance (Y) component only, and then convert back to RGB for display. This is because: i) processing R, G, and B components independently may alter the color balance, and ii) the human visual system is not very sensitive to high frequencies in chrominance components. Hence, this chapter considers processing of monochrome images only.

## 3.1 Image Smoothing

Image smoothing refers to removing high-frequency details, which yields a softer or somehow blurry image. It is often employed as a pre-processing or an intermediate processing step in many image-processing operations, including image decimation and interpolation, gradient estimation, and image enhancement. It is also used for image denoising. Indeed, the algorithms discussed in Section 3.5 are image-smoothing algorithms tailored to particular noise models. We can classify image-smoothing algorithms as linear shift-invariant (LSI) filters, and nonlinear/adaptive filters.

### 3.1.1 Linear Shift-Invariant Low-Pass Filtering

A low-pass filtered image $s_L(n_1, n_2)$ can be computed either in the discrete Fourier transform DFT domain in terms of the filter-frequency response (see Chapter 1), or in the spatial domain by 2D-convolution summation

$$s_L(n_1,n_2) = \sum\sum_{(i_1,i_2)\in W} h(i_1,i_2)\; s(n_1 - i_1, n_2 - i_2) \tag{3.1a}$$

where $h(i_1, i_2)$ denotes the impulse response of the filter and $W$ is the filter support. The impulse response must be normalized such that

$$\sum\sum_{(i_1,i_2)\in W} h(i_1,i_2) = 1 \tag{3.1b}$$

so that the mean intensity of the filtered image remains unchanged.

Often separable or circularly symmetric filters (see Chapter 1) are preferred for ease of design and implementation. The most popular 2D linear shift-invariant smoothing filters are uniform (box) and Gaussian filters, whose impulse responses are depicted in Figure 3.1. Both filters are separable and can be derived from the 1D box filter

$$h(n) = [1\ 1\ 1\ 1\ 1]$$

and the 1D Gaussian filter

$$h(n) = [1\ 4\ 7\ 4\ 1]$$

respectively. The frequency responses of these 1D filters are shown in Figure 3.2.

| 1 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 |

(a)

| 1 | 4 | 7 | 4 | 1 |
|---|---|---|---|---|
| 4 | 16 | 28 | 16 | 4 |
| 7 | 28 | 49 | 28 | 7 |
| 4 | 16 | 28 | 16 | 4 |
| 1 | 4 | 7 | 4 | 1 |

(b)

**Figure 3.1** 5 × 5 filter kernels with $N_1 = N_2 = 2$: (a) box filter and (b) Gaussian filter with $\sigma = 1$. The coefficients need to be normalized so that their sum is equal to 1.

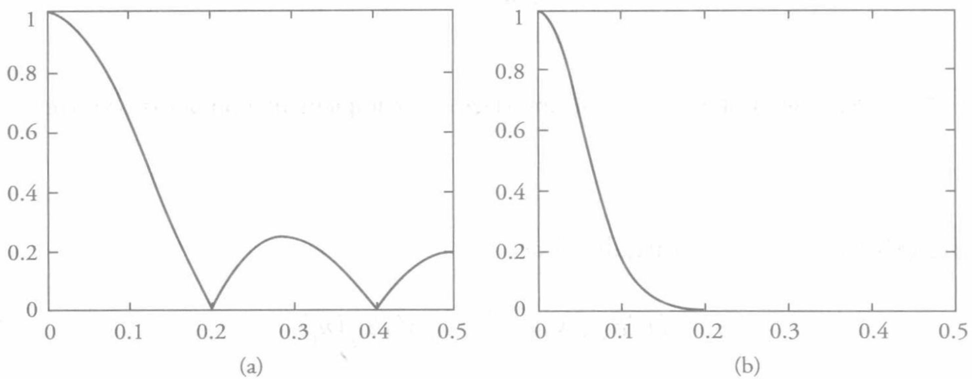



**Figure 3.2** Frequency response of (a) 5 × 1 box filter and (b) Gaussian filter with $\sigma = 3$.

## Box Filtering

If $h(n_1, n_2)$ is uniform over a $(2N_1 + 1)\times(2N_2 + 1)$ rectangular support, it can be implemented by a fast computational algorithm, called box filtering [McD 81]. The implementation of box filtering requires two running buffers, a vertical sum buffer (VSB) and a horizontal sum buffer (HSB), which are shown in Figure 3.3. The VSB is updated once every line by removing one whole line (the topmost line) and adding one new line. The HSB computes a running average over the current VSB. The VSB is initialized as

$$VSB_{N_2}(n_1) = \sum_{n_2=0}^{2N_2} s(n_1, n_2) \tag{3.2a}$$

and is updated for $n_2 = N_2, \ldots$ as

$$VSB_{n_2+1}(n_1) = VSB_{n_2}(n_1) - s(n_1, n_2 - N_2) + s(n_1, n_2 + N_2 + 1) \tag{3.2b}$$

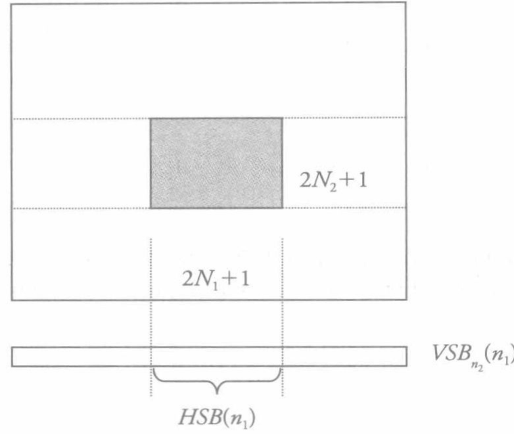



**Figure 3.3** Illustration of the VSB and HSB for the implementation of the box filter.

The HSB for line $n_2$ is initialized as

$$HSB_{n_2}(N_1) = \sum_{n_1=0}^{2N_1} VSB_{n_2}(n_1) \tag{3.3a}$$

and is updated for $n_1 = N_1, \ldots$ as

$$HSB_{n_2}(n_1+1) = HSB_{n_2}(n_1) - VSB_{n_2}(n_1 - N_1) + VSB_{n_2}(n_1 + N_1 + 1) \tag{3.3b}$$

Then, the filtered output image is given by

$$s_L(n_1, n_2) = \frac{1}{(2N_1+1)(2N_2+1)} HSB_{n_2}(n_1) \tag{3.4}$$

Hence, the computational complexity of box filtering is equal to one multiplication (division), two additions, and two subtractions per pixel, independent of the size of the filter impulse response (kernel).

### Gaussian Filtering

The Marr-Hildreth scale space theory employs a Gaussian kernel, with the scale parameter $\sigma$, which can be implemented as a finite-impulse response filter over a $(2N+1)\times(2N+1)$ support given by

$$h(n_1, n_2) = K\, e^{-\left(\frac{n_1^2+n_2^2}{2\sigma^2}\right)} \quad -N \le n_1 \le N, -N \le n_2 \le N \tag{3.5a}$$

where $K$ is a normalization constant. The scale parameter $\sigma$ specifies the amount of smoothing to be applied; the larger the $\sigma$, the more the smoothing. A typical value is $\sigma = 2$. It can be seen that the Gaussian filter evaluates a weighted sum of input pixels according to their distance from the center pixel. Note that $h(n_1, n_2)$ is separable, where

$$h(n_1,n_2) = K\,e^{-\left(\frac{n_1^2+n_2^2}{2\sigma^2}\right)} = C\,e^{-\left(\frac{n_1^2}{2\sigma^2}\right)} \cdot C\,e^{-\left(\frac{n_2^2}{2\sigma^2}\right)} = h_1(n_1)\cdot h_2(n_2) \tag{3.5b}$$

Hence, $h_1(n_1)$ and $h_2(n_2)$ can be applied to row and columns of the image, respectively, for efficient implementation. In order to implement the filter using integer arithmetic, the value of $C$ is set so that the smallest coefficient in $h_1(n_1)$ or $h_2(n_2)$ is equal to 1. All other coefficients are truncated to the nearest integer. The filter output is normalized such that (3.1b) is satisfied.

### 3.1.2 Bi-Lateral Filtering

LSI filters, such as the Gaussian filter, assume local similarity; i.e., pixel intensities in nearby locations are similar to each other. However, this assumption breaks down near edges, where contrast and/or color of pixels change suddenly. As a result, modeling similarity by geometric proximity (in the domain space of an image) causes blurring and/or color bleeding artifacts. Several nonlinear/adaptive filtering approaches, including directional filtering, anisotropic diffusion, rank-order filtering, mean-shift (which converges to the local mode), and bi-lateral filtering, have been proposed to overcome this well-known problem. Among these filters, bi-lateral filtering, which can be considered as an adaptive extension of Gaussian filtering, has become ubiquitous in image-processing and computer graphics applications including multi-resolution image representations, tone mapping, denoising, and texture editing/relighting [Par 08]. A theoretical analysis and discussion of the relationship between bi-lateral filtering and other nonlinear/adaptive image-filtering frameworks can be found in [Bar 04, Par 08].

In bi-lateral filtering, similarity is modeled by both geometric proximity and photometric similarity (distance between pixel intensities, i.e., in the range space of an image). The name bi-lateral filtering originates from this combined domain and range-space filtering [Tom 98]. The basic idea of bi-lateral filtering (sometimes called sigma filtering or robust filtering) is to compute a weighted average of pixels $s(\mathbf{k})$ in a local $(2N+1)\times(2N+1)$ neighborhood of the current pixel $\mathbf{n}$, which are within some gray-level distance of the intensity of current (center) pixel $s(\mathbf{n})$, given by

$$g(\mathbf{n}) = \textstyle\sum_{\mathbf{k}} \mathrm{w}(\mathbf{n},\mathbf{k})\; s(\mathbf{k}) \tag{3.6}$$

where $\mathbf{n} = (n_1, n_2)^T$, $\mathbf{k} = (k_1, k_2)^T$ and the weights $w(\mathbf{n}, \mathbf{k})$ depend on both spatial proximity and photometric similarity between pixels $\mathbf{n}$ and $\mathbf{k}$.

In bi-lateral Gaussian filtering, both the spatial-closeness function $p(\cdot)$ and the intensity similarity function $q(\cdot)$ are Gaussian. More specifically,

$$p(\mathbf{n} - \mathbf{k}) = e^{-\frac{\|\mathbf{n}-\mathbf{k}\|^2}{2\sigma_p^2}} \tag{3.7a}$$

where $\|\cdot\|$ denotes the Euclidean distance of pixel $\mathbf{k}$ from the center pixel $\mathbf{n}$ of an $N \times N$ kernel, and $\sigma_p^2$ determines the importance of spatial proximity, while

$$q(\mathbf{n} - \mathbf{k}) = e^{-\frac{d(s(\mathbf{n}),\, s(\mathbf{k}))}{2\sigma_q^2}} \tag{3.7b}$$

where $d(s(\mathbf{n}), s(\mathbf{k})) = \|s(\mathbf{n}) - s(\mathbf{k})\|^2$ is the distance between the intensity or color of pixels $\mathbf{n}$ and $\mathbf{k}$ and $\sigma_q^2$ denotes the importance of intensity similarity. The coefficients (weights) of the bi-lateral filter when centered at pixel $\mathbf{n}$ are given by

$$w(\mathbf{n}, \mathbf{k}) = K(\mathbf{n})\, p(\mathbf{n} - \mathbf{k})\; q(\mathbf{n} - \mathbf{k}) \tag{3.7c}$$

where $K(\mathbf{n})$ is a normalization constant so the weights for all pixels sum to one.

In summary, each pixel is replaced by a weighted average of its neighbors where the weights are computed as in (3.7c). There are three free parameters: the size of the local neighborhood $N$, the scale parameter $\sigma_p^2$, and the range parameter $\sigma_q^2$. When the range parameter is large, the bi-lateral filter approaches the Gaussian filter. Otherwise, when the bi-lateral filter is centered on the bright side of an edge, the similarity function $q(\cdot)$ takes values close to one for pixels on the same side, and close to zero for pixels on the dark side of the edge. As a result, the filter replaces the center pixel by an average of bright pixels in its vicinity. Conversely, when the filter is centered on a dark pixel, a weighted average of only darker pixels are computed, which helps preserve edges.

Several fast computational methods, based on the idea of quantization along the intensity axis and down-sampling in the spatial domain, have been proposed for efficient implementation of the bi-lateral filter [Par 08].

## 3.2 Image Re-Sampling and Multi-Resolution Representations

Image re-sampling, also known as decimation and interpolation, requires evaluation of image intensity at sub-pixel locations. It appears in many image-processing

problems including image scaling, color de-mosaicking, multi-resolution representations, sub-pixel motion estimation, motion-compensated filtering, image warping, and synthetic view synthesis. The filters used for image decimation and interpolation have a significant effect on the quality of the results. Most image re-sampling filters are separable; hence, 1D filters are applied independently in $n_1$ and $n_2$ directions. This reduces image re-sampling to a 1D-signal-re-sampling problem.

In multi-rate digital signal processing, decimation and interpolation are used to match the sampling rate of a signal with the bandwidth requirements of a specific application [Cro 83]. Interpolation refers to the process of up-sampling followed by appropriate filtering, while decimation refers to appropriate filtering followed by down-sampling. In the following, we first discuss decimation and then interpolation by an integer factor. Sampling rate change by a rational factor and polyphase filtering for efficient implementation are also presented. We present multi-resolution (pyramid and wavelet) image representation as an application of image re-sampling.

## 3.2.1 Image Decimation

Decimation refers to down-sampling of a signal; hence, it technically can only be applied to over-sampled signals (in the sense of the Nyquist sampling rate) without loss of information. Otherwise, it either causes aliasing (if no anti-alias filter is applied) or blurring (if the proper anti-alias filtering is applied). Decimation by a factor of $M$ can be modeled in two steps: first, multiplication by an impulse train to replace $M-1$ samples in between every $M$th sample with zeros, and then discarding the zero samples to obtain a signal at the lower rate. We describe these steps in the following.

Given the input signal $s(n)$, define an intermediate signal $w(n)$ by

$$w(n) = s(n)\sum_{k=-\infty}^{\infty} \delta(n-kM) \tag{3.8}$$

Then, the signal decimated by a factor of $M$ can be expressed as

$$y(n) = w(Mn) \tag{3.9}$$

The decimation process in the spatial domain without anti-alias filtering is illustrated in Figure 3.4 for $M=2$. The intermediate signal $w(n)$ is introduced to facilitate the frequency-domain characterization of decimation. To this effect, we first compute the Fourier transform of the intermediate signal $w(n)$ using (3.8) as

$$W(e^{j\omega}) = \sum_{n=-\infty}^{\infty} \left\{ s(n)\sum_{k=-\infty}^{\infty} \delta(n-kM) \right\} e^{-j\omega n} = \frac{1}{M}\sum_{k=0}^{M-1} S\left( e^{j\left(\omega - \frac{2\pi k}{M}\right)} \right) \tag{3.10}$$

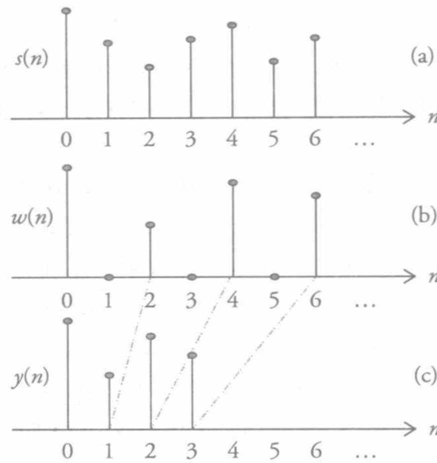



**Figure 3.4** Decimation by $M = 2$: (a) input signal; (b) intermediate signal; and (c) down-sampled signal.

It can be seen that the spectrum of the intermediate signal has $M$ replications of the input signal spectrum in the interval $(-\pi, \pi)$. The spectrum $Y(e^{j\omega})$ of the final decimated signal is obtained by expansion of the frequency axis of $W(e^{j\omega})$ given by

$$Y(e^{j\omega}) = \sum_{n=-\infty}^{\infty} w(Mn)\, e^{-j\omega n} = W\left(e^{j\frac{\omega}{M}}\right) = \frac{1}{M}\sum_{k=0}^{M-1} S\left(e^{j\left(\frac{\omega-2\pi k}{M}\right)}\right) \tag{3.11}$$

If the bandwidth of the input $S(e^{j\omega})$ is more than $\pi/M$, then the replications will overlap and the decimated signal will suffer from aliasing. This is expected, because the sampling rate of the decimated signal should not be allowed to fall below the Nyquist rate. If an application mandates going below the Nyquist rate, then appropriate anti-alias filtering should be applied prior to decimation. Ideal anti-alias filtering requires an ideal low-pass filter, as shown in Figure 3.5. The cutoff frequency of the ideal anti-alias filter is $\pi/M$ for decimation by $M$.

Common choices for realizable anti-alias filters are box, Gaussian (Section 3.1.1), or bi-lateral filters (Section 3.1.2). Box filtering averages all pixels within a local window. Although it is a poor approximation to the ideal low-pass filter, it is often preferred because of computational simplicity. The Gaussian and bi-lateral filters are employed in multi-resolution image representations.

### Efficient Polyphase Implementation of Decimation Filters

We apply anti-alias filtering to the input image before down-sampling (at the high rate); however, $M-1$ out of every $M$ samples of the filtered image are discarded

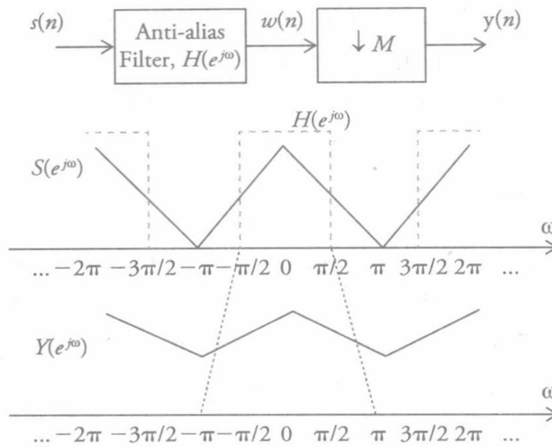



**Figure 3.5** Decimation by $M = 2$: (a) system diagram; (b) spectrum of the input and the anti-alias filter; and (c) spectrum of the down-sampled output signal.

during the down-sampling step. This redundancy may be avoided by skipping the computation of samples that will be discarded in the standard serial implementation, but that is not very efficient since the arithmetic units would remain idle in $M-1$ sampling periods out of every $M$. An efficient implementation can be obtained based on the following observations: i) each retained output sample is generated by a sub-set of the filter coefficients, and ii) the subset of filter coefficients for each output sample varies in a periodic pattern with period $M$. Polyphase implementation exploits these facts by forming $M$ parallel branches, one for each subset of filter coefficients, and the order of anti-alias filtering and down-sampling is interchanged at each branch. That is, each branch works at low (output) rate, and the outputs of the $M$ parallel short filters are summed to form the desired decimated signal. Details of polyphase implementation of decimation filters can be found in [Mit 06].

## 3.2.2 Interpolation

Interpolation refers to computing sub-pixel intensity values. Image-interpolation methods can be classified as LSI vs. adaptive/nonlinear filters. The LSI interpolation process can be analyzed in two steps: i) up-sampling by zero filling (also called "filling-in"), and ii) low-pass filtering of the zero-filled signal. We first characterize the frequency spectrum of the zero-filled signal. Given a signal $s(n)$, we define a signal $u(n)$ that is upsampled (zero-filled) by $L$ as

$$u(n) = \begin{cases} s\left(n/L\right) & n = 0, \pm L, \pm 2L, \ldots \\ 0 & \text{otherwise} \end{cases} \tag{3.12}$$

The process of up-sampling by zero filling is demonstrated in Figure 3.6 for the case $L=3$. Next, we take the Fourier transform of $u(n)$ given by Eqn. (3.12). Using the definition of the Fourier transform for discrete-time signals, we have

$$U(e^{j\omega}) = \sum_{n=-\infty}^{\infty} u(n)\, e^{-j\omega n} = \sum_{n=-\infty}^{\infty} s(n)\, e^{-j\omega Ln} = S(e^{j\omega L}) \tag{3.13}$$

It can be seen from (3.13) that the spectrum of the zero-filled signal is related to the spectrum of the input signal by a compression of the frequency axis. This is illustrated in Figure 3.7 for the case of $L=3$. Note that the spectrum of the input $S(e^{j\omega})$ is assumed to occupy the full bandwidth, which is the interval $(-\pi, \pi)$ since the Fourier transform of discrete signals are periodic, with the period equal to $2\pi$ along the normalized frequency axis.

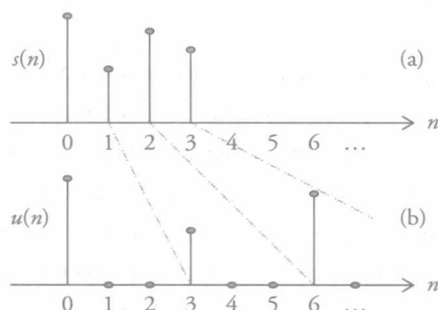



**Figure 3.6** Up-sampling by $L=3$: (a) input signal and (b) zero-filled signal.

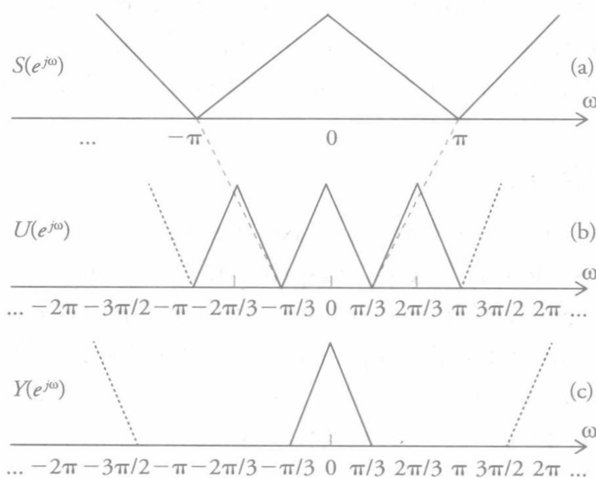



**Figure 3.7** Spectra of signals: (a) input; (b) zero-filled $L=3$; and (c) after ideal filtering.

Interpolation filtering aims at eliminating the replications caused by the zero-filling process. This requires ideal low-pass filtering of the filled-in signal as shown in Figure 3.7. The ideal low-pass filter would have a DC gain factor of $L$ and cutoff frequency of $\omega_c = \pi/L$.

In the time domain, the filtering operation can be viewed as replacing the zero samples with non-zero values by means of a smoothing operation. The impulse response of the ideal interpolation filter is a sinc function given by

$$h(n) = \frac{\sin(\pi n / L)}{\pi n / L} \tag{3.14}$$

Thus, the interpolated signal samples are given by

$$y(n) = \sum_{k=-\infty}^{\infty} s(k) \frac{\sin(\pi(n-k)/L)}{\pi(n-k)/L} \tag{3.15}$$

The impulse response of the ideal LSI interpolation filter has the properties that $h(0) = 1$ and $h(n) = 0$, for $n = \pm L, \pm 2L, \ldots$. Because of these zero crossings, $y(n) = s(n)$ at the existing sample values, while assigning non-zero values for the zero samples in the upsampled signal. This *sample preservation property* is an important characteristic of all interpolation filters.

The ideal interpolation filter is unrealizable since it is an infinite-impulse response, non-causal filter; i.e., its implementation requires infinite-time delay. Thus, it is approximated by one of the following realizable filters, which also possess the sample preservation property.

### Zero-Order Hold Filter

Zero-order hold is the simplest interpolation method that corresponds to pixel replication. The impulse response of the zero-order hold filter is given by

$$h(n) = \begin{cases} 1 & \text{if } 0 \le n \le L-1 \\ 0 & \text{otherwise} \end{cases} \tag{3.16}$$

and its implementation is depicted in Figure 3.8.

Note that the zero-order hold filter is a poor interpolation filter, since its frequency response is given by a sinc function,

$$H(e^{j\omega}) = \sum_{n=0}^{L-1} e^{-j\omega n} = \frac{1-e^{-j\omega L}}{1-e^{-j\omega}} = e^{-j\omega\frac{(L-1)}{2}} \frac{\sin\left(\omega L/2\right)}{\sin\left(\omega/2\right)} \tag{3.17}$$

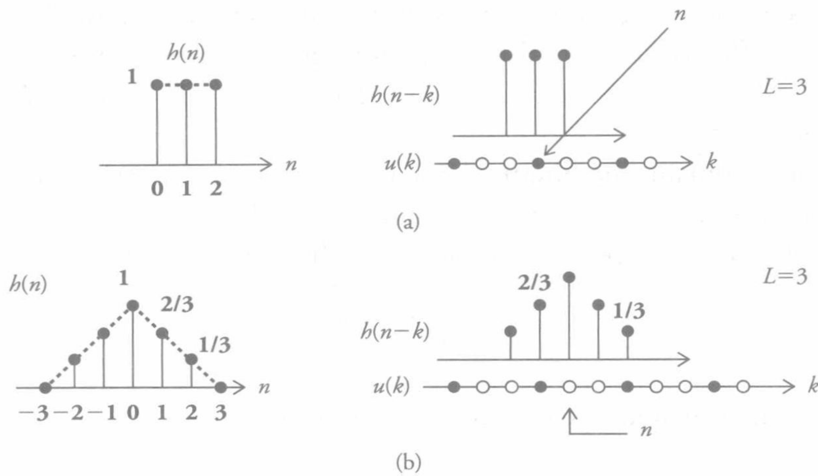



**Figure 3.8** Interpolation filters ($L=3$): (a) zero-order hold and (b) linear interpolation.

which has large sidelobes. These sidelobes prevent effective removal of replications in the frequency domain, which results in aliasing artifacts in the interpolated signal.

### Linear Interpolation

Linear interpolation computes a weighted sum of two nearest neighbor pixels (one on each side). The weights are inversely proportional to the distance of the pixel to be interpolated from its neighbors, resulting in unequal weights if $L \neq 2$. The impulse response of the linear interpolation filter is given by

$$h(n) = \begin{cases} \dfrac{L-|n|}{L} & \text{if } 0 \leq |n| \leq L-1 \\ 0 & \text{otherwise} \end{cases} \tag{3.18}$$

which is shown in Figure 3.8 for $L=3$. The frequency response of the linear interpolation filter is equal to the square of the sinc function. Thus, it has lower sidelobes than the zero-order hold filter. The filter (3.18) has the sample preserving property.

### Cubic-Convolution Interpolation

The cubic-convolution filter computes a weighted sum of four nearest neighbor pixels (two on each side). The weights are obtained by approximating the impulse response of the ideal low-pass filter in Eqn. (3.14) by a piecewise cubic polynomial consisting of three cubics.

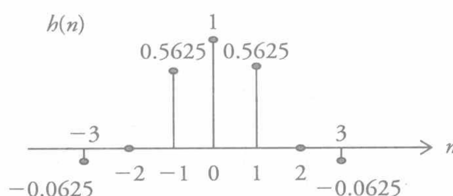



**Figure 3.9** Impulse response of the cubic-convolution filter for $L=2$.

A method for designing finite-impulse response (FIR) cubic-convolution filters has been proposed by Keys [Key 81] (see Exercise 3.2). The impulse response of the filter, which has $4L-1$ taps, is demonstrated in Figure 3.9 for $L=2$. It is straightforward to show that this filter also has the sample preserving property. We note that the cubic convolution filter approximates the unrealizable ideal low-pass filter better than a truncated sinc filter of the same length, because the frequency response of the truncated sinc filter suffers from ringing due to the well-known Gibbs phenomenon [Mit 06].

### Efficient Polyphase Implementation of Interpolation

The polyphase implementation is an efficient implementation that avoids multiplications by zeros. The subsets of the impulse response coefficients that affect computation of each of $L$ samples define the $L$ polyphase components of the interpolation filter [Mit 06]. The input $s(n)$ is fed into each of the polyphase filters without zero-filling (at the low rate) as depicted in Figure 3.10. The output samples from component filters are interleaved in sequential order to form the interpolated output (high rate).

### Sampling Rate Change by a Rational Factor

The theory of decimation and interpolation (by an integer factor) easily extends to re-sampling by a rational factor $L/M$, by first interpolating by a factor $L$ and then decimating the result by a factor of $M$. Since an interpolator and a decimator are cascaded, the interpolation (post-) filter and anti-alias (pre-) filter of decimation can be merged into a single low-pass filter with the cutoff frequency $f_c=\min\{\pi/M, \pi/L\}$, which is depicted in Figure 3.11. The filter must satisfy the constraint that when location of the filter output sample matches an existing input sample location, the values of the existing samples must be preserved. If $M > L$, the system effectively performs decimation; otherwise, it performs interpolation.

Re-sampling with a fractional factor $L/M$ means that $L$ output samples will fall at non-integer locations between $M$ existing samples (pixels), whose values must be preserved. The polyphase implementation of re-sampling with a fractional factor

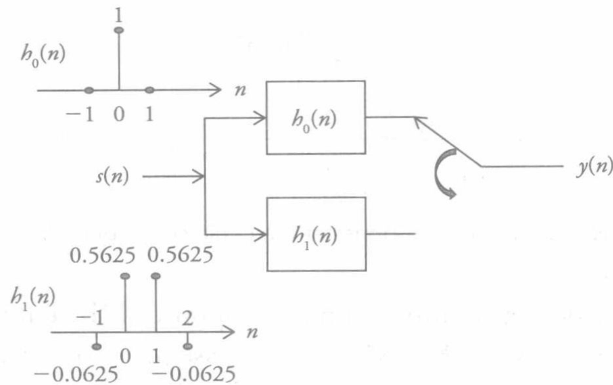



**Figure 3.10** Polyphase implementation of the cubic-convolution filter in Figure 3.9.

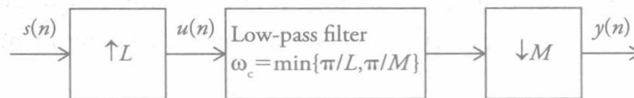



**Figure 3.11** Sampling rate change by a factor of $L/M$.

$L/M$ requires $L$ component filters that are fed by input $s(n)$, similar to that shown in Figure 3.10, whose outputs are sampled sequentially to be placed at these non-integer locations. The polyphase implementation may result in significant computational savings, especially for prime factors, where the numerator is large.

### Adaptive/Nonlinear Interpolation and Single-Frame Super-Resolution

LSI filters produce the desired number of output pixels but cannot generate higher resolution, i.e., cannot create details that are not present in the original image (frequencies beyond $\pi/L$ are zero in Figure 3.7). Interpolation methods that aim to recover frequencies higher than $\pi/L$ from a single low-resolution image are called *single-frame super-resolution* (SR) methods. This is an ill-posed problem; hence, the solution must rely on some image model (see Appendix A). Te solutions can be classified as adaptive/nonlinear interpolation filters and single-frame SR methods.

#### *Adaptive/Nonlinear Interpolation*

LSI filtering treats all pixels the same. In contrast, adaptive/nonlinear filters change orientation and/or filter kernel depending on the local image properties to render edges and texture with higher fidelity and minimize interpolation artifacts where they are most apparent. Edge-adaptive filters consist of two steps [Wan 07]:

i) detect the presence, orientation, and energy of an edge (see Section 3.3), and ii) interpolate by adapting the shape and parameters of the filter according to edge orientation and energy. Li and Orchard [Li 01] proposed an edge-directed interpolation algorithm for natural images. They use local covariance coefficients, estimated from the low-resolution image, to adapt the interpolation filter by exploiting the geometric duality between the low-resolution and high-resolution covariances so that the filter coefficients are tuned to match an arbitrarily oriented step edge. A hybrid approach of switching between bilinear interpolation and covariance-based adaptive interpolation is proposed to reduce the overall computational complexity. Alternatively, anisotropic Gaussian filtering, where the filter kernel is tuned according to local edge orientation similar to bi-lateral filtering, has been proposed [Han 13]. A data-adaptive regression kernels framework unifies steerable filters and bi-lateral filtering [Tak 07]. Among stochastic model-based methods, Schultz and Stevenson [Sch 94] modeled the high-resolution image by a discontinuity-preserving Huber-Markov random field model and computed its maximum *a posteriori* (MAP) estimate.

*Single-Frame SR*

Single-frame SR methods include machine-learning methods and sparse model-based filters. Learning methods aim to establish a mapping from low-resolution image patches to high-resolution patches either using a set of pre-computed example pairs (dictionary) or self-similarity. Then, each input low-resolution image patch is compared to a set of stored low-resolution patches, and the high-resolution patch corresponding to the nearest low-resolution patch satisfying neighborhood matching criteria (e.g., a Markov network model) is selected as the output. Examples of this approach include the hallucination method [Bak 02], example-based super-resolution [Fre 02], and the sparse regression method [Kim 10].

Sparse modeling (see Appendix A) has proven useful for super-resolution, where a low-resolution image is modeled by a blur matrix $\mathbf{H}$ and sub-sampling matrix $\mathbf{L}$, as

$$\mathbf{y} = \mathbf{L}\mathbf{H}\,\mathbf{s} + \mathbf{v} \tag{3.19}$$

In the case of image interpolation, the blur matrix $\mathbf{H}$ is taken as the identity matrix. Hence, sparse modeling becomes less effective since the data consistency term fails to constrain the local image structure. To this effect, Dong *et al.* [Don 13a] proposed incorporating non-local image self-similarity into sparse image representation by using a non-local auto-regressive model as the data fidelity term.

**Color De-Mosaicking**

Color de-mosaicking, used in most color cameras today, is an important application of image interpolation. Color filter arrays (CFA) for color image capture were discussed in Section 2.2.2 in Chapter 2. De-mosaicking refers to interpolation of missing pixels in each color channel. Commonly used bilinear and bicubic filters often cause false colors when applied to each color channel independently due to aliasing errors. To this effect, specific methods that take the correlation between R, G, and B channels into account have been developed for CFA interpolation [Gun 05, Men 11]. The interchannel correlation is modeled by assuming that color ratios (or differences) for an object remain constant, which prevents abrupt changes in hue. In order to exploit this model, the G channel is interpolated first, since it has the most pixels, using an edge-adaptive filter. Then $R/G$ and $B/G$ images are formed for existing $R$ and $B$ pixels, which are then independently interpolated. Final $R$ and $B$ images are obtained by multiplying the interpolated ratio images by the $G$ channel values.

## 3.2.3 Multi-Resolution Pyramid Representations

In a multi-resolution pyramid representation, depicted in Figure 3.12, the original $N_1 \times N_2$ image $s_0(n_1, n_2)$ appears at the bottom (level 0). This image is decimated (low-pass filtered and sub-sampled) by a factor of two (2) in each direction. The resulting $N_1/2 \times N_2/2$ image $s_1(n_1, n_2)$ appears at the first level of the pyramid. The procedure can be repeated until a desired number of levels is reached. If a low-pass filter with a truncated Gaussian impulse response is used, the resulting pyramid is known as a Gaussian pyramid.

The Gaussian pyramid is an overcomplete (redundant) representation since the total number of pixels in the pyramid

$$\sum_{l=0}^{\infty} \frac{N_1 N_2}{4^l} = N_1 N_2 + \frac{N_1 N_2}{4} + \frac{N_1 N_2}{16} + \cdots = \frac{4}{3} N_1 N_2$$

approaches 4/3 times the number of pixels in the image in the limit as the number of levels goes to infinity. Since the frequency response of the Gaussian filter has some leakage beyond the frequency $\omega = \pi/2$, images in the upper levels may contain aliasing. In some vision applications, the sub-sampling step may be skipped such that all images are the same size but successively more blurred to avoid aliasing artifacts in lower resolution (upper-level) images.

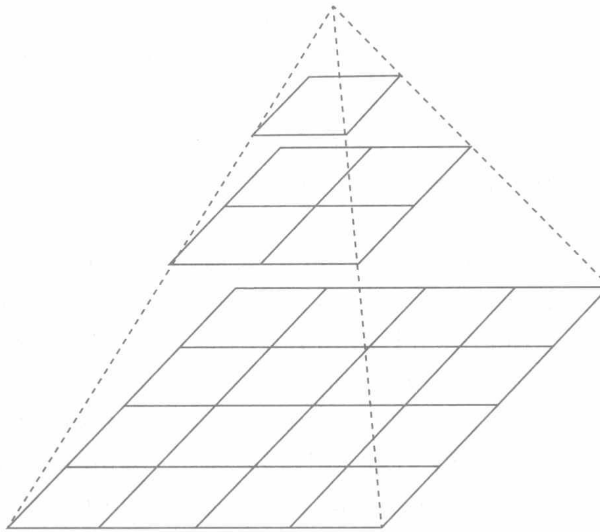


**Figure 3.12** Multi-resolution (multi-scale) pyramid representation.

### 3.2.4 Wavelet Representations

Unlike the pyramid representation, wavelet transfom is a multi-resolution image representation that is critically sampled, i.e., the number of samples in the wavelet transform is equal to that of the original image. It is preferred not only for image coding but also for other filtering applications, such as image denoising. There are many classes of wavelet transforms with different properties, such as compact support (using FIR filters), orthogonality, symmetry, regularity, and degree of smoothness. Two of the most popular classes for image processing/compression are the orthogonal and bi-orthogonal classes of wavelet transforms.

In order to understand discrete wavelet analysis, we consider the two-channel (binary) decomposition, where a 1D signal $s(n)$ is split into two equal-size frequency bands, called the lower and upper frequency bands, as shown in Figure 3.13.

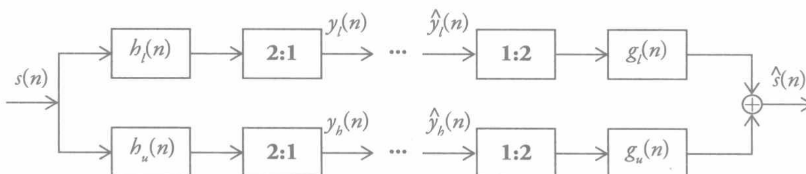



**Figure 3.13** Block diagram of sub-band decomposition and reconstruction filtering.

In wavelet analysis, the scaling function $\Phi$ and mother wavelet $\Psi$ can be associated with a low-pass filter $H_l(f)$ and a high-pass filter $H_u(f)$, respectively. If we let the normalized sampling frequency equal to $\omega = 2\pi$ or $f=1$, where $\omega = 2\pi f$, theoretically, we need an ideal low-pass filter with the passband $f \in (0,1/4)$ and an ideal high-pass filter with the passband $f \in (1/4,1/2)$ for binary decomposition. In practice, we employ FIR filters, with impulse responses $h_l[n]$ and $h_u[n]$, subject to some constraints that are discussed below. The outputs of these analysis filters are sub-sampled by 2 to obtain the low-pass and high-pass subsignals, $y_l[n]$ and $y_h[n]$, respectively. Note that, due to sub-sampling, the sum of the number of samples in $y_l[n]$ and $y_h[n]$ is equal to the number of samples in $s(n)$, hence, the wavelet transform is critically sampled. After processing or compression/decompression in the wavelet domain, sub-signals $\hat{y}_l[n]$ and $\hat{y}_h[n]$ are upsampled by zero filling, then filtered using filters $g_l[n]$ and $g_u[n]$, whose outputs are summed to reconstruct the signal $\hat{s}(n)$. The filters $g_l[n]$ and $g_u[n]$ are called synthesis filters.

The analysis-synthesis filters should have the following desired properties:

1. Perfect-reconstruction (PR): PR refers to designing filters $h_l[n]$, $h_u[n]$, $g_l[n]$, and $g_u[n]$ such that forward transform (analysis filtering) followed by inverse transform (synthesis filtering) gives $\hat{s}(n) = s(n)$, assuming that subsignals are not altered in the wavelet domain, i.e., $\hat{y}_l[n] = y_l[n]$ and $\hat{y}_h[n] = y_h[n]$. In order to design realizable filters, we have to allow an overlap between the passbands of the low-pass and high-pass filters to avoid any frequency gaps. A pair of low-pass and high-pass filters, whose frequency responses exhibit mirror symmetry about $f=1/4$, as depicted in Figure 3.14, is called a quadrature mirror filter (QMF) pair. The fact that the frequency response of the low-pass filter $H_l(f)$ extends into the band $(1/4,1/2)$ and vice versa causes unavoidable aliasing when each sub-band signal is decimated by 2.

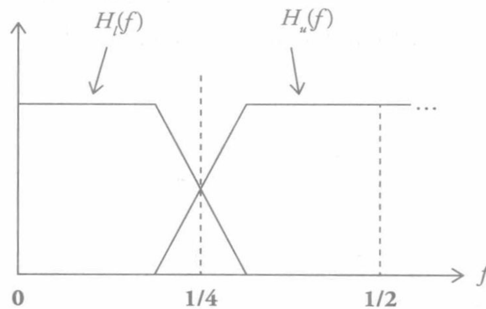



**Figure 3.14** Frequency response of 1D binary decomposition filters.

In order to achieve alias-free (PR) analysis-synthesis filtering, the filters must be designed in such a way that the aliasing introduced by the analysis filter is exactly canceled by the synthesis filter. In order to see how this can be achieved, we express the Fourier transform of the subsignals $y_l[n]$ and $y_h[n]$, after subsampling by 2, as

$$Y_l(f) = \frac{1}{2}\left[H_l\left(\frac{f}{2}\right)S\left(\frac{f}{2}\right) + H_l\left(-\frac{f}{2}\right)S\left(-\frac{f}{2}\right)\right] \tag{3.20a}$$

$$Y_h(f) = \frac{1}{2}\left[H_u\left(\frac{f}{2}\right)S\left(\frac{f}{2}\right) + H_u\left(-\frac{f}{2}\right)S\left(-\frac{f}{2}\right)\right] \tag{3.20b}$$

respectively. The reconstructed signal can be expressed as

$$\hat{S}(f) = G_l(f)\hat{Y}_l(2f) + G_u(f)\hat{Y}_h(2f) \tag{3.21}$$

Assuming $\hat{Y}_l(f) = Y_l(f)$ and $\hat{Y}_h(f) = Y_h(f)$, and substituting (3.20) into (3.21),

$$\hat{S}(f) = \frac{1}{2}[H_l(f)G_l(f) + H_u(f)G_u(f)]\ S(f)$$
$$+\frac{1}{2}[H_l(-f)G_l(f) + H_u(-f)G_u(f)]\ S(-f)$$

Alias-cancellation (PR) can be achieved provided the filters satisfy

$$H_l(f)G_l(f) + H_u(f)G_u(f) = 2 \tag{3.22a}$$

$$H_l(-f)G_l(f) + H_u(-f)G_u(f) = 0 \tag{3.22b}$$

2. Symmetry: An important concern in image transforms is avoiding increasing the number of samples. Since linear convolution increases the number of samples, filtering is implemented by circular convolution, which yields the same number of output samples as that of the input. We apply a symmetric boundary extension in order to avoid introducing unnecessary high-frequency energy due to artificial left-to-right and top-to-bottom intensity discontinuities. However, to preserve symmetry after filtering so the number of wavelet coefficients does not increase, the filters must also be symmetric. Note that odd- (even-) length symmetric FIR filters are zero- (linear) phase filters.

3. Orthogonality: Orthogonal filters implement a projection of the input image onto a set of orthogonal basis functions. With proper normalization, orthogonal transforms preserve energy and norm, as stated by Parseval's theorem.

It has been shown that the only filter that satisfies two-channel perfect reconstruction, symmetry, and orthogonality conditions is the trivial case of 2-tap Haar filter pair, $h_l[n] = \{1,1\}$ and $h_u[n] = \{1,-1\}$, which does not have good frequency selectivity or energy compaction properties. Hence, in order to design perfect reconstruction FIR filters (compactly supported wavelets) with good frequency selectivity or energy compaction (a larger number of vanishing moments), we need to give up either symmetry or orthogonality. It turns out we can design *orthogonal* FIR filters that are perfect reconstruction and nearly symmetric or that are symmetric and nearly perfect reconstruction, or *bi-orthogonal* (non-orthogonal) symmetric FIR filters that enable perfect reconstruction. In the following, we briefly discuss the classes of orthogonal and bi-orthogonal filters used in wavelet image processing and compression in more detail.

**Orthogonal Filters**

Early sub-band coding methods have employed orthogonal, symmetric FIR (linear phase) quadrature mirror filters (QMF) that are nearly perfect reconstruction. Orthogonal, FIR perfect reconstruction filters, such as Symlet or Coiflet families that are nearly symmetric [Dau 92], are preferred for image denoising, since orthogonal transform of white Gaussian noise (image domain) is again white Gaussian in the wavelet domain.

In orthogonal QMF design, perfect reconstruction can be achieved by canceling the aliased spectra, given by (3.22b), with the following simple choice of filters:

$$\begin{aligned} H_u(f) &= H_l(-f) = H_l\left(f + \frac{1}{2}\right) \\ G_l(f) &= 2\, H_l(f) \\ G_u(f) &= -2\, H_u(f) \end{aligned} \qquad (3.23)$$

Note that the first condition is equivalent to

$$h_u[n] = (-1)^n h_l[n]$$

Hence, orthogonal FIR analysis and synthesis filters have the same length. Substituting (3.23) into (3.21), the reconstructed signal becomes

$$\hat{S}(f)=\left[H_l^2(f)-H_u^2(f)\right]S(f)$$

For perfect reconstruction, it follows that orthogonal filters must also satisfy

$$\left[H_l^2(f)-H_u^2(f)\right]=1 \text{ for all } f \tag{3.24}$$

Johnston filters allow for some small amplitude distortion in (3.24), while Symlets and Coiflets are near symmetric PR filters designed to satisfy some vanishing moments and regularity constraints. A complete derivation of these filters is beyond the scope of this book, and interested readers are referred to [Dau 92] for details.

### Bi-Orthogonal Filters

In image compression, orthogonality decorrelates the transform coefficients to minimize redundancy and ensures the energy of the quantization error committed by quantization of the transform coefficients remains unchanged in the pixel domain. However, symmetry turns out to be more important than orthogonality, since symmetry is crucial for properly handling image borders without increasing the number of transform samples. Hence, bi-orthogonal filters have become the *de facto* choice for wavelet image compression. The bi-orthogonality constraints enable designing perfect reconstruction, symmetric (linear phase) FIR filters by relaxing the strict orthogonality requirement. The perfect reconstruction conditions (3.22a) and (3.22b) can also be stated as

$$H_l(f)G_l(f)+H_l(-f)G_l(-f)=2 \tag{3.25a}$$

$$H_u(f)G_u(f)+H_u(-f)G_u(-f)=2 \tag{3.25b}$$

$$H_l(f)G_u(f)+H_l(-f)G_u(-f)=0 \tag{3.25c}$$

$$H_u(f)G_l(f)+H_u(-f)G_l(-f)=0 \tag{3.25d}$$

which can be expressed in the spatial domain as bi-orthogonality constraints [Dau 92]

$$\langle h_l[k], g_l[2n-k]\rangle=\delta[n] \tag{3.26a}$$

$$\langle h_u[k], g_u[2n-k]\rangle=\delta[n] \tag{3.26b}$$

$$\langle g_u[k], h_l[2n-k]\rangle=0 \tag{3.26c}$$

$$\langle g_l[k], h_u[2n-k]\rangle=0 \tag{3.26d}$$

where $\langle \cdot,\cdot \rangle$ denotes the inner product. In wavelet terms, the filters $\{h_l[n], h_u[n]\}$ are derived from a pair $\{\Phi_1, \Psi_1\}$, while the filters $\{g_l[n], g_u[n]\}$ are derived from another pair $\{\Phi_2, \Psi_2\}$ which are related to $\{\Phi_1, \Psi_1\}$ by the bi-orthogonality constraints. The bi-orthogonality conditions provide us with more flexibility to design odd-length symmetric FIR filters.

There is a large family of bi-orthogonal wavelets. Among these, (9,7) filters are nearly orthogonal and provide good energy compaction. The (9,7) and (5,3) filters have been selected for use in the JPEG2000 standard. The bi-orthogonal wavelet filters possess some regularity properties that orthogonal filters do not have, which provides bi-orthogonal filters with improved coding efficiency over orthogonal filters with the same number of taps (see Chapter 7).

The 1D decompositions can be extended to two dimensions by using separable filters, i.e., decomposing the image $s(n_1, n_2)$ first in the row and then in the column direction, or vice versa. Using a binary decomposition in each direction, we obtain four sub-bands called low (L) $y_L(n_1, n_2)$, horizontal (H) $y_H(n_1, n_2)$, vertical (V) $y_V(n_1, n_2)$, and diagonal (D) $y_D(n_1, n_2)$, corresponding to lower-lower, upper-lower, lower-upper, and upper-upper subbands, respectively. The decomposition can be continued by splitting all sub-bands or just the L-subband repetitively, as shown in Figure 3.15. Other decompositions are also possible.

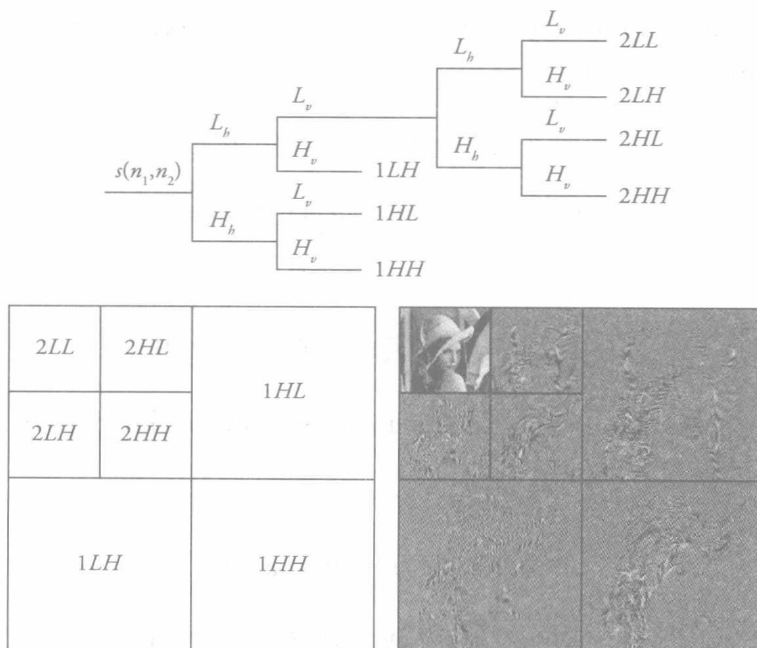



**Figure 3.15** Two-level binary-tree decomposition of an image into frequency bands.

The wavelet transform coefficients then correspond to pixels of the respective subimages. In most cases, the decomposition is carried out in multiple stages. The total number of samples in all subimages $y_L(n_1, n_2)$, $y_V(n_1, n_2)$, $y_H(n_1, n_2)$, and $y_D(n_1, n_2)$ after subsampling is the same as the number of samples in the input image $s(n_1, n_2)$. Thus, the wavelet decomposition itself does not result in data compression or expansion. Observe that $y_L(n_1, n_2)$ corresponds to a low-resolution version of the image $s(n_1, n_2)$, while $y_D(n_1, n_2)$ contains the high-frequency detail information. Therefore, the wavelet decomposition is also known as a "multi-scale" or "multi-resolution" representation, and can be used in progressive transmission.

## 3.3 Image-Gradient Estimation, Edge and Feature Detection

In order to understand modeling image edges, we first look at a continuous 1D signal $s(x)$, where an ideal edge can be represented by a unit-step function. However, in most real-life signals transition from low to high or from high to low intensity value is gradual and the edge location can be defined as the point where the first derivative $s'(x)$ has an extremum, or equivalently, where the second derivative $s''(x)$ is zero, as shown in Figure 3.16.

Images are two-dimensional; hence, we need to replace the concepts of first and second derivative with the gradient vector and Laplacian, respectively. The gradient vector for a continuous image $s_c(x_1, x_2)$ is defined as

$$\nabla s_c(x_1, x_2) = \begin{bmatrix} \dfrac{\delta s_c(x_1, x_2)}{\delta x_1} \\ \dfrac{\delta s_c(x_1, x_2)}{\delta x_2} \end{bmatrix} \tag{3.27}$$

where $\delta s_c / \delta x$ denotes partial derivatives. The magnitude of the gradient is given by

$$|\nabla s_c(x_1, x_2)| = \sqrt{\left(\frac{\delta s_c(x_1, x_2)}{\delta x_1}\right)^2 + \left(\frac{\delta s_c(x_1, x_2)}{\delta x_2}\right)^2} \tag{3.28}$$

The Laplacian of a continuous image is the dot product of its gradient by itself

$$\nabla^2 s_c(x_1, x_2) = \nabla \cdot \nabla s_c(x_1, x_2) = \frac{\delta^2 s_c(x_1, x_2)}{(\delta x_1)^2} + \frac{\delta^2 s_c(x_1, x_2)}{(\delta x_2)^2} \tag{3.29}$$

which is a scalar. The Laplacian is isotropic favoring no particular edge direction.

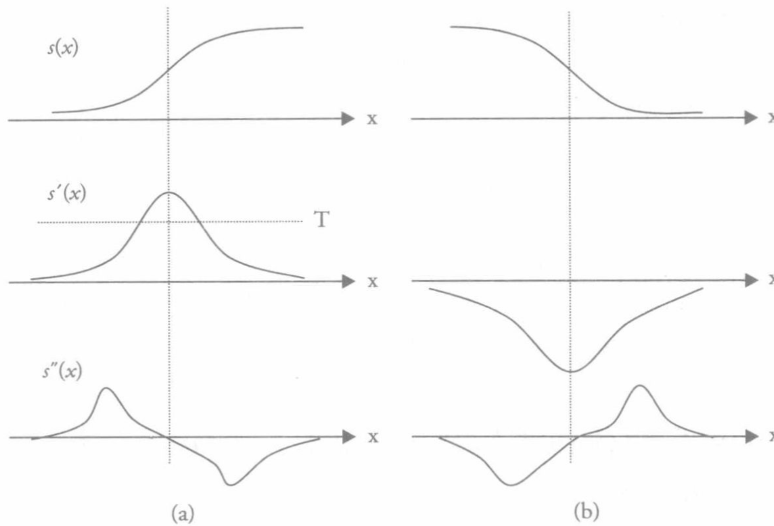



**Figure 3.16** Illustration of an edge for a 1D continuous signal and its first and second derivatives: (a) increasing edge and (b) decreasing edge.

Next, we need to estimate the first and second partials from discrete images, which are discussed in Section 3.3.1 and 3.3.2, respectively. Because the derivative operators are indeed high-pass filters, edge detection is sensitive to noise. Two types of errors may be observed: i) False positives: Noise may generate many small peaks in the magnitude of the gradient resulting in false edges. ii) False negatives: Noise may result in shifts in true edge locations, resulting in missing actual edge pixels. A popular edge-detection algorithm that addresses these problems to find meaningful edges is introduced in Section 3.3.3.

## 3.3.1 Estimation of the Image Gradient

We first discuss approximating partial derivatives by finite differences and provide some popular so-called "edge-detection operators" based on these approximations, and then present a method for estimating partials by the derivative of Gaussian filter.

### Estimation of Partials by Finite-Difference Operators

Given a discrete image $s(n_1, n_2)$, the horizontal and vertical partial derivatives can be approximated by respective finite differences. The horizontal partial can be estimated by the horizontal forward difference

$$\frac{\delta s_c(x_1, x_2)}{\delta x_1} \approx s[n_1 + 1, n_2] - s[n_1, n_2] \tag{3.30a}$$

or the horizontal backward difference

$$\frac{\delta s_c(x_1, x_2)}{\delta x_1} \approx s[n_1, n_2] - s[n_1 - 1, n_2] \tag{3.30b}$$

Since we do not know which one will be a better estimate, we can compute the average of the forward- and backward-finite differences, called the central difference, as a more robust estimate

$$\frac{\delta s_c(x_1, x_2)}{\delta x_1} \approx \frac{1}{2}\left(s[n_1 + 1, n_2] - s[n_1 - 1, n_2]\right) \tag{3.31}$$

Finite differences are sensitive to noise. In order to alleviate the effects of observation noise, we can compute a local average (at the same horizontal sample $n_1$ over the current line $n_2$, the line before and the line after) of the average differences, called the average central difference

$$\begin{aligned}\frac{\delta s_c(x_1, x_2)}{\delta x_1} \approx s_{x_1}(n_1, n_2) = \frac{1}{6}\{&\left(s[n_1 + 1, n_2] - s[n_1 - 1, n_2]\right) \\ &+\left(s[n_1 + 1, n_2 - 1] - s[n_1 - 1, n_2 - 1]\right) \\ &+\left(s[n_1 + 1, n_2 + 1] - s[n_1 - 1, n_2 + 1]\right)\}\end{aligned} \tag{3.32}$$

Other averaging strategies also exist for estimating partials using finite differences. Estimation of the partials in the vertical direction can be treated in a similar way.

Hence, the gradient of a discrete image can be expressed as

$$\nabla s_c(x_1, x_2) \approx \nabla s[n_1, n_2] = \begin{bmatrix} s_{x_1}(n_1, n_2) \\ s_{x_2}(n_1, n_2) \end{bmatrix} = \begin{bmatrix} h_1(n_1, n_2) ** s(n_1, n_2) \\ h_2(n_1, n_2) ** s(n_1, n_2) \end{bmatrix} \tag{3.33}$$

where the computation of $s_{x_1}(n_1, n_2)$ and $s_{x_2}(n_1, n_2)$ can be interpreted as 2D-convolution operations, i.e., FIR filtering of an image $s(n_1, n_2)$ with finite-impulse responses $h_1(n_1, n_2)$ and $h_2(n_1, n_2)$, respectively, or 2D-correlation operations. Since 2D convolution requires flipping the impulse response with respect to both axes, correlation operator kernels are 2D flipped versions of the impulse responses.

Since the gradient is a vector, the magnitude and the direction of the gradient vector at a pixel $(n_1, n_2)$, which indicate the strength and the direction of a possible edge at the pixel $(n_1, n_2)$, respectively, are given by

$$\left|\nabla s[n_1,n_2]\right| = \sqrt{\left(s_{x_1}(n_1,n_2)\right)^2 + \left(s_{x_2}(n_1,n_2)\right)^2} \tag{3.34a}$$

$$\measuredangle\left(\nabla s[n_1,n_2]\right) = \tan^{-1}\left(\frac{s_{x_2}(n_1,n_2)}{s_{x_1}(n_1,n_2)}\right) \tag{3.34b}$$

In the following, we introduce some commonly used operators for gradient estimation, which are also known as edge-detection operators. The FIR filters $h_1(n_1, n_2)$ and $h_2(n_1, n_2)$ are given by 2D flipped versions of these operators.

The *Prewitt operator* represents the average central difference approximation given by (3.32). The Prewitt kernels for estimation of partials in the $x_1$ and $x_2$ directions are shown in Figure 3.17(a) and (b). The more popular horizontal and vertical *Sobel operators* are shown in Figure 3.17(c) and (d). The difference between Prewitt and Sobel operators is that the latter applies twice the weight to the center row horizontally and vertically. Isotropic filters do not favor any particular edge direction. Prewitt and Sobel filters respond to diagonal edges differently than the horizontal and vertical edges because their coefficients do not take into account larger inter-pixel distances in the diagonal directions. The Prewitt filter is less sensitive to diagonal edges than to horizontal and vertical ones, while the opposite is true for the Sobel filter. The *Roberts cross operators,* shown in Figure 3.17(e) and (f), aim to

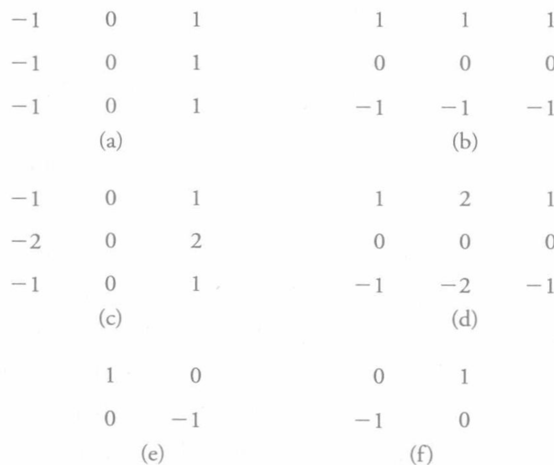



**Figure 3.17** Operators to compute image partials, also called edge-detection operators: (a) horizontal Prewitt operator; (b) vertical Prewitt operator; (c) horizontal Sobel operator; (d) vertical Sobel operator; (e) and (f) Roberts cross operators.

approximate the gradient of an image by computing the sum of squares of the differences between diagonally adjacent pixels.

**Estimation of Partials by Derivatives of Gaussian Filtering**

Spatial presmoothing of an image with a Gaussian filter usually helps with gradient estimation in the presence of noise. The estimation of partial derivatives from Gaussian smoothed images can be modeled as

$$s_{x_1}(n_1,n_2) = h_1(n_1,n_2) ** \left(g(n_1,n_2) ** s(n_1,n_2)\right) \tag{3.35a}$$

where $g(n_1, n_2)$ is the Gaussian smoothing filter (see Section 3.1.2) and $h_1(n_1, n_2)$ is a finite-difference operator to compute the partial in the horizontal direction. Since 2D convolution is associative, we can rewrite (3.35a) as

$$s_{x_1}(n_1,n_2) = \left(h_1(n_1,n_2) ** g(n_1,n_2)\right) ** s(n_1,n_2) \tag{3.35b}$$

The combined filter is called the derivative of Gaussian filter, given by

$$\tilde{h}_1(n_1,n_2) = h_1(n_1,n_2) ** g(n_1,n_2) \tag{3.36}$$

In order to evaluate the impulse response of the derivative of Gaussian filter, we take a continuous 2D Gaussian with scale parameter $\sigma$

$$g_c(x_1,x_2) = Ke^{-\left(\frac{x_1^2+x_2^2}{2\sigma^2}\right)}$$

and compute its partials in the horizontal and vertical directions as

$$\tilde{h}_1(x_1,x_2) = \frac{\delta g_c(x_1,x_2)}{\delta x_1} = -K\frac{x_1}{\sigma^2}e^{-\left(\frac{x_1^2+x_2^2}{2\sigma^2}\right)} \tag{3.37a}$$

$$\tilde{h}_2(x_1,x_2) = \frac{\delta g_c(x_1,x_2)}{\delta x_2} = -K\frac{x_2}{\sigma^2}e^{-\left(\frac{x_1^2+x_2^2}{2\sigma^2}\right)} \tag{3.37b}$$

Implementation of the derivative of Gaussian filtering requires sampling $\tilde{h}_1(x_1,x_2)$ and $\tilde{h}_2(x_1,x_2)$ over a finite support for a given value of $\sigma$ (see Exercise 3.1).

We note that the derivative of the Gaussian filter is separable; hence, both the horizontal and vertical partial filters can be efficiently implemented as a cascade of two 1D filters each, where

$$\tilde{h}_1(n_1,n_2)=\left\{-K_1\frac{n_1}{\sigma^2}e^{-\left(\frac{n_1^2}{2\sigma^2}\right)}\right\}\left\{K_2 e^{-\left(\frac{n_2^2}{2\sigma^2}\right)}\right\} \tag{3.38a}$$

and

$$\tilde{h}_2(n_1,n_2)=\left\{K_1 e^{-\left(\frac{n_1^2}{2\sigma^2}\right)}\right\}\left\{-K_2\frac{n_2}{\sigma^2}e^{-\left(\frac{n_2^2}{2\sigma^2}\right)}\right\} \tag{3.38b}$$

Computation of image gradients is often implemented over a multi-resolution Gaussian pyramid, shown in Figure 3.12, in top-to-bottom fashion. Edges at the top (coarser) levels of the pyramid correspond to major edges, while other edges at lower (finer) levels are regarded as finer details.

## 3.3.2 Estimation of the Laplacian

The Laplacian is a scalar, which is the counterpart of the second derivative for multi-variable functions. It can be estimated by finite-difference operators or by the Laplacian of Gaussian filtering [Hue 86]. Recall that the zero-crossings of a Laplacian-filtered image indicate edge locations.

### Estimation by Finite-Difference Operators

In order to compute a discrete approximation to the Laplacian, we use the forward difference to approximate the first horizontal partial derivative:

$$s_{x_1}[n_1,n_2]=s[n_1+1,n_2]-s[n_1,n_2]$$

and then the backward difference of first differences, to approximate the second derivative as the derivative of the first derivative, to compute the second horizontal difference $s_{x_1x_1}[n_1,n_2]$ as follows:

$$\begin{aligned} s_{x_1x_1}[n_1,n_2] &= s_{x_1}[n_1,n_2]-s_{x_1}[n_1-1,n_2] \\ &= s[n_1+1,n_2]-2s[n_1,n_2]+s[n_1-1,n_2] \end{aligned} \tag{3.39a}$$

The second vertical difference is computed similarly, given by

$$s_{x_2x_2}[n_1,n_2]=s[n_1,n_2+1]-2s[n_1,n_2]+s[n_1,n_2-1] \tag{3.39b}$$

| 0 | 1 | 0 |
|---|---|---|
| 1 | −4 | 1 |
| 0 | 1 | 0 |

(a)

| 1 | 1 | 1 |
|---|---|---|
| 1 | −8 | 1 |
| 1 | 1 | 1 |

(b)

| −1 | 2 | −1 |
|---|---|---|
| 2 | −4 | 2 |
| −1 | 2 | −1 |

(c)

**Figure 3.18** Different approximations yield different Laplacian operators.

Then, a discrete approximation to the Laplacian can be defined as

$$\nabla^2 s(x_1,x_2) \approx s_{x_1x_1}[n_1,n_2] + s_{x_2x_2}[n_1,n_2] \tag{3.40}$$
$$= s[n_1+1,n_2] + s[n_1-1,n_2] + s[n_1,n_2+1] + s[n_1,n_2-1] - 4s[n_1,n_2]$$

Eqn. (3.40) can be considered as an FIR filter with the impulse response shown in Figure 3.18(a). Other approximations to estimate the Laplacian yield other FIR filters, shown in Figure 3.18(b) and (c).

### Estimation by the Laplacian of the Gaussian Filter

Similar to the derivation of the derivative of the Gaussian filter, we can merge pre-smoothing of the image by a Gaussian filter and estimation of the Laplacian into a single filter, given by

$$\nabla^2\left[s_c(x_1,x_2) ** g_c(x_1,x_2)\right] = \nabla^2\left[g_c(x_1,x_2)\right] ** s_c(x_1,x_2) \tag{3.41}$$

since both the convolution and Laplacian are linear shift invariant operations. Hence, we can define the Laplacian of Gaussian (LoG) filter as

$$h_c(x_1,x_2) = \nabla^2\left[g_c(x_1,x_2)\right] = K\frac{x_1^2+x_2^2-2\sigma^2}{\sigma^4}e^{-\left(\frac{x_1^2+x_2^2}{2\sigma^2}\right)} \tag{3.42}$$

Digital implementation of the LoG filter requires sampling $h_c(x_1,x_2)$ on an appropriate support for a particular value of $\sigma$. We note that the LoG filter is not separable, but it can be approximated by a difference of Gaussians (DoG) filter

$$\nabla^2\left[g_c(x_1,x_2)\right] \approx K\left[e^{-\left(\frac{x_1^2+x_2^2}{2\sigma_1^2}\right)} - e^{-\left(\frac{x_1^2+x_2^2}{2\sigma_2^2}\right)}\right] \tag{3.43}$$

with proper choice of $\sigma_1$ and $\sigma_2$ for efficient implementation.

### 3.3.3 Canny Edge Detection

We treat edge detection for a monochrome image $s(n_1, n_2)$ only, since edge detection for color images is often performed on the luminance (Y) channel. Edge pixels are defined as those pixels where the magnitude of the gradient has a maximum and/or the Laplacian is zero (see Figure 3.16). Determination of edge pixels is often implemented by simple thresholding as follows:

$$\left|\nabla s(n_1,n_2)\right| > T \tag{3.44a}$$

and/or

$$\nabla^2 s(n_1,n_2) < \varepsilon \tag{3.44b}$$

where $T$ and $\varepsilon$ are some threshold values [Dav 75, Sha 01]. The problem with simple thresholding as in Eqn. (3.44) is in how to determine good threshold values ($T$ and $\varepsilon$) and an appropriate scale parameter $\sigma$ to be used in the derivative of Gaussian and LoG filters. If $\sigma$ is large, the image will be smoothed (blurred) too much and some edges may be lost. If $T$ is chosen too small or $\varepsilon$ is too big (in their own scales), then detected edges may be too thick (poor localization); otherwise, some edges may not be detected (poor detection).

Canny's edge detection, perhaps the most widely used method, addresses a compromise between good detection and good localization. Canny [Can 86] shows that the derivative of the Gaussian filter provides a close approximation to an optimal filter that maximizes the product of a localization and detection measure. Canny's method also includes a non-local maxima suppression step for thinning, and a hysterisis thresholding step for computing meaningful connected edges.

The complete Canny edge-detection procedure is as follows:

1. *Computation of Image Gradient:* Evaluate the magnitude and direction of the image gradient, given by (3.34a) and (3.34b), at each pixel, where $s_{x_1}(n_1, n_2)$ and $s_{x_2}(n_1, n_2)$ are computed by the derivative of Gaussian filters (3.38a) and (3.38b), respectively.

2. *Non-Local Maximum Suppression:* This is an edge-thinning procedure using the direction and magnitude of the gradient. If the magnitude of the gradient at the center pixel $(n_1, n_2)$ of an 8-neighborhood is less than the magnitude of the gradient in at least one of its two neighbors in the direction of the gradient, the magnitude of the gradient at the pixel $(n_1, n_2)$ is set equal to 0. The computed gradient direction is rounded off to one of 0, 45, 90, or 135 degrees to determine the two 8-neighbors of the center pixel $(n_1, n_2)$ for comparison.
3. *Hysterisis Thresholding:* This is an edge-linking procedure. Two thresholds, a high and a low threshold, are defined for edge detection and edge following, respectively, where the high threshold is two or three times the low threshold. This is a two-step procedure: i) All pixels where the magnitude of the gradient is above the high threshold are labeled as edge pixels, and ii) all pixels where the magnitude of the gradient exceeds the lower threshold are kept as edge pixels if they are connected to an already labeled edge pixel.

A set of edge maps over a range of scales can be produced by varying $\sigma$. A fine-to-coarse synthesis can fuse edges at different scales into a single edge map.

### 3.3.4 Harris Corner Detection

Consider taking an image patch centered at the pixel $(x_1, x_2)$ and shifting it by $(d_1, d_2)$. The weighted sum of squared differences (SSD) between the original and shifted patches is given by

$$E(d_1,d_2) = \sum_{x_1}\sum_{x_2} w(x_1,x_2)[s(x_1+d_1,x_2+d_2) - s(x_1,x_2)]^2$$

If we approximate $s(x_1+d_1, x_2+d_2)$ by a Taylor expansion

$$s(x_1+d_1,x_2+d_2) = s(x_1,x_2) + s_{x_1}(x_1,x_2)\, d_1 + s_{x_2}(x_1,x_2)\, d_2$$

where $s_{x_1}(x_1,x_2)$ and $s_{x_2}(x_1,x_2)$ denote partial derivatives of $s(x_1,x_2)$, we can express the SSD

$$E(d_1,d_2) \approx \sum_{x_1}\sum_{x_2} w(x_1,x_2)[s_{x_1}(x_1,x_2)\, d_1 + s_{x_2}(x_1,x_2)\, d_2]^2$$

which can be rewritten in matrix form as

$$E(d_1,d_2) \approx \begin{bmatrix} d_1 & d_2 \end{bmatrix} \mathbf{M} \begin{bmatrix} d_1 \\ d_2 \end{bmatrix}$$

where

$$\mathbf{M} = \begin{bmatrix} \sum_{x_1}\sum_{x_2} w(x_1,x_2)(s_{x_1}(x_1,x_2))^2 & \sum_{x_1}\sum_{x_2} w(x_1,x_2)s_{x_1}(x_1,x_2)s_{x_2}(x_1,x_2) \\ \sum_{x_1}\sum_{x_2} w(x_1,x_2)s_{x_1}(x_1,x_2)s_{x_2}(x_1,x_2) & \sum_{x_1}\sum_{x_2} w(x_1,x_2)(s_{x_2}(x_1,x_2))^2 \end{bmatrix}$$

is called the Harris matrix. If a circular weighting, such as a Gaussian, is used, then the response will be isotropic. The Harris matrix is a function of first partials of an image about a patch $(x_1, x_2)$.

At a corner or an interest point, the function $E(d_1, d_2)$ must exhibit a large variation for all shifts $(d_1, d_2) \neq (0,0)$. This condition can be expressed as **M** should have two "large" eigenvalues at an interest point. We can reach the following conclusions based on the magnitudes of the eigenvalues:

1. If $\lambda_1 \approx 0$ and $\lambda_2 \approx 0$, then the pixel $(x_1, x_2)$ has no features of interest.
2. If $\lambda_1 \approx 0$ and $\lambda_2 \gg \lambda_1$ a horizontal edge, or $\lambda_2 \approx 0$ and $\lambda_1 \gg \lambda_2$ a vertical edge, is found.
3. If $\lambda_1$ and $\lambda_2$ both have large positive values greater than a threshold, then a corner is found.

Harris and Stephens [Har 88] observe that exact computation of eigenvalues is computationally expensive, and instead suggest the following cornerness function:

$$C = \lambda_1\lambda_2 - \alpha(\lambda_1 + \lambda_2)^2 = \det \mathbf{M} - \alpha(\text{trace }(\mathbf{M}))^2$$

where $\alpha$ is a tunable sensitivity parameter. Their algorithm does not actually compute the eigenvalues of the matrix **M**; instead, they evaluate the determinant and trace of **M** to find corners. The Shi–Tomasi [Shi 94] corner detection method (also known as detection of good features to track) checks $\min\{\lambda_1, \lambda_2\}$ because the eigenvalue analysis produces more stable corners for motion tracking.

Scale-invariant feature transform (SIFT ) keypoints are another set of corner-like features that are computed by analyzing the output of DoG filters at successive levels of scale [Low 04]. The SIFT system uses a feature post-processing stage, which is similar to that of the Harris detector.

## 3.4 Image Enhancement

Image enhancement refers to processing with the goal of improving attributes, such as contrast and sharpness, of an image to make it visually more pleasing. It has received significant attention for more than four decades since the era of film-based photography. This section addresses two specific problems: i) given an image with poor contrast, compute a better-looking image with higher contrast and sharper details; ii) given a high dynamic range image (acquired as described in Chapter 2), apply dynamic range compression to better render details in bright or dark areas on a standard display. Image-enhancement methods can be classified as pixel-based contrast-enhancement and spatial-filtering methods.

1. *Pixel-based tone mapping/contrast enhancement:* Underexposed or overexposed pictures have poor contrast. Contrast stretching (also known as histogram normalization) is a process that scales pixel intensity (brightness) values to better utilize the range between 0 and 255. It is sometimes called dynamic-range expansion; however, this is not technically correct since dynamic range refers to the resolution of image-brightness values, and pixel-based contrast-enhancement operators cannot generate new intermediate pixel values to increase the brightness resolution or image detail. Pixel-based contrast-enhancement methods are discussed in Section 3.4.1.
2. *Spatial filtering for tone mapping and image sharpening:* Linear or nonlinear spatial filtering can be applied to emphasize medium-range spatial frequencies to obtain crispier images. These filters can also be combined with pixel-based operators to implement tone mapping for dynamic-range compression of high-dynamic range (HDR) images to display them on a standard 8 bits/color display systems. Spatial filters are discussed in Section 3.4.2.

### 3.4.1 Pixel-Based Contrast Enhancement

Pixel-based contrast-enhancement operators map all pixels with a given input brightness value to the same output brightness value, which corresponds to shifting and

scaling of the independent variable (intensity) of the histogram of an image. We start by defining the histogram of an image. We then introduce commonly used pixel-based contrast-enhancement operations.

**Image Histogram**

Assume that gray values of $s(n_1, n_2)$ are quantized to $K$ levels, i.e., $0 \leq s(n_1,n_2) \leq K-1$ for all $(n_1, n_2)$. The histogram $H_s(k)$ gives the relative frequency of occurrence of each gray-level $k$ in the image. That is,

$$H_s(k) = \frac{J}{N_1 N_2} \quad \text{for} \quad k = 0,\ldots,K-1 \tag{3.45}$$

if the gray-level $k$ occurs $J$ times in the image and we have an $N_1 \times N_2$ image. The pixel counts $J$ are normalized by the total number of pixels $N_1 N_2$ in the image, so that

$$\sum_{k=0}^{K-1} H_s(k) = 1$$

This way, the histogram $H_s(k)$ approximates the probability density function (pdf) of the image. That is, if a pixel location $(n_1, n_2)$ is chosen at random, then $H_s(k)$ gives the probability that $s(n_1,n_2) = k$. Statistics of the image can be computed from its histogram. For example, the mean value of an image can be computed as

$$\mu = \sum_{k=0}^{K-1} k \; H_s(k) \tag{3.46}$$

We can also define the cumulative normalized image histogram as

$$C_s(k) = \sum_{i=0}^{k} H_s(i), \;\; k = 0,\ldots,K-1 \tag{3.47}$$

which approximates the cumulative distribution function (CDF) of the image. It means that for a randomly selected pixel $(n_1, n_2)$, $\Pr\{s(n_1, n_2) \leq k\} = C_s(k)$. We note that the CDF is a non-decreasing function, and $C_s(K-1) = 1$. Furthermore, $H_s(k)$ can be obtained from $C_s(k)$ by

$$H_s(k) = C_s(k) - C_s(k-1), \;\; k = 0,\ldots,K-1$$

### Linear Contrast Manipulation

Suppose we process an input image $s(n_1, n_2)$ using the generalized linear mapping

$$g(n_1,n_2)=\alpha\ s(n_1,n_2)+\beta \tag{3.48}$$

If we let $\alpha=1$, then we have an additive bias. This corresponds to shifting the histogram to the left or right depending on the sign of $\beta$. Of course, proper clipping should be used to ensure pixel values do not overshoot or undershoot the allowable range. If $\alpha \neq 1$, then we also have a contraction or stretching of the histogram depending on whether $\alpha < 1$ or $\alpha > 1$. Again, proper clipping should be used to ensure pixel values do not overshoot or undershoot the allowed range.

#### Example 1: Automatic Gain Control (AGC)

Many digital cameras employ AGC that stretch the image histogram to fully utilize the entire dynamic range from 0 to $K-1$. Let the minimum and maximum gray levels in an image be $A$ and $B$, respectively. The goal of AGC is to find the mapping parameters to map $A$ and $B$ to $0$ and $K-1$, respectively, such that

$$\alpha A+\beta=0$$

$$\alpha B+\beta=K-1$$

which can be solved for the two unknowns to give

$$\alpha=\frac{K-1}{B-A}$$

$$\beta=-A\left(\frac{K-1}{B-A}\right)$$

Hence, the AGC mapping is given by

$$g(n_1,n_2)=\frac{K-1}{B-A}\,(s(n_1,n_2)-A)$$

#### Example 2: Image Negative

The negative of an image can be computed as

$$g(n_1,n_2)=-\,s(n_1,n_2)+(K-1)$$

which corresponds to flipping of the image histogram

$$H_g(k) = H_s(K-1-k)$$

**Histogram Equalization**

Histogram equalization is a pixel-based nonlinear operation that aims to produce an output image with a more uniform histogram. An image with a flat histogram has maximum entropy. This is achievable if $H_s(\cdot)$ and $C_s(\cdot)$ are functions of continuous variables (intensity values). Digital histogram equalization maps every occurrence of a quantized gray level $k$ to $C_s(k)$; hence, histogram bins can never be increased or reduced in pixel count. Therefore, it is not possible to attain an output image with a perfectly flat histogram. However, it is possible to obtain an output image, which has a more strecthed out histogram than the input image. In the following, we first derive the histogram equalization operation assuming that $H_s(\cdot)$ and $C_s(\cdot)$ are functions of continuous variables. We then discuss the digital approximation of it.

Suppose that $H_s(x)$ and $C_s(x)$ are functions of a continuous variable $x$, such that

$$H_s(x) = \frac{dC_s(x)}{dx}$$

where $C_s(x)$ is non-decreasing. We assume that an inverse $C_s^{-1}(x)$ can be defined for $C_s(x)$ since $C_s(x)$ is non-decreasing. Then, we claim that the image

$$g(n_1,n_2) = C_s(s(n_1,n_2)) \tag{3.49}$$

has a uniform (flat) histogram. We can show this as follows:

$$\begin{aligned} C_g(x) &= Pr\{g(n_1,n_2) \leq x\} = Pr\{C_s(s[n_1,n_2]) \leq x\} \\ &= Pr\{s[n_1,n_2] \leq C_s^{-1}(x)\} = C_s(C_s^{-1}(x)) = x,\ 0 \leq x \leq 1 \end{aligned}$$

Therefore,

$$H_g(x) = \frac{dC_g(x)}{dx} = 1,\ 0 \leq x \leq 1$$

Based on this analysis, a procedure for digital histogram equalization can be summarized as:

1. Compute the histogram $H_s(k)$ of the image, $k = 0, \ldots, K-1$
2. Compute the cumulative distribution function

$$C_s(k) = \sum_{i=0}^{k} H_s(i),\ k = 0, \ldots, K-1$$

3. Compute $g[n_1, n_2] = (K-1)\ C_s(s[n_1, n_2])$ for all pixels $(n_1, n_2)$

**Histogram Shaping**

Histogram shaping is a generalization of histogram equalization, where the output image should have a pre-specified desired histogram as opposed to having a flat histogram. Again, the exact desired histogram can only be obtained in the hypothetical case of continuous-intensity gray-scale images. Suppose the desired CDF is $Q(x)$ and $Q^{-1}(x)$ can be defined. Then,

$$g(n_1, n_2) = Q^{-1}(C_s(s(n_1, n_2))) \tag{3.50}$$

has the desired CDF $Q(x)$ if every pixel has continuous (not-quantized) intensity values. This can be shown, similar to the case of histogram equalization, as

$$\begin{aligned}
C_g(x) &= Pr\{g(n_1, n_2) \leq x\} = Pr\{Q^{-1}(C_s(s(n_1, n_2))) \leq x\} \\
&= Pr\{C_s(s(n_1, n_2)) \leq Q(x)\} \\
&= Pr\left\{s(n_1, n_2) \leq C_s^{-1}(Q(x))\right\} = C_s\left(C_s^{-1}(Q(x))\right) = Q(x),\ 0 \leq x \leq 1
\end{aligned}$$

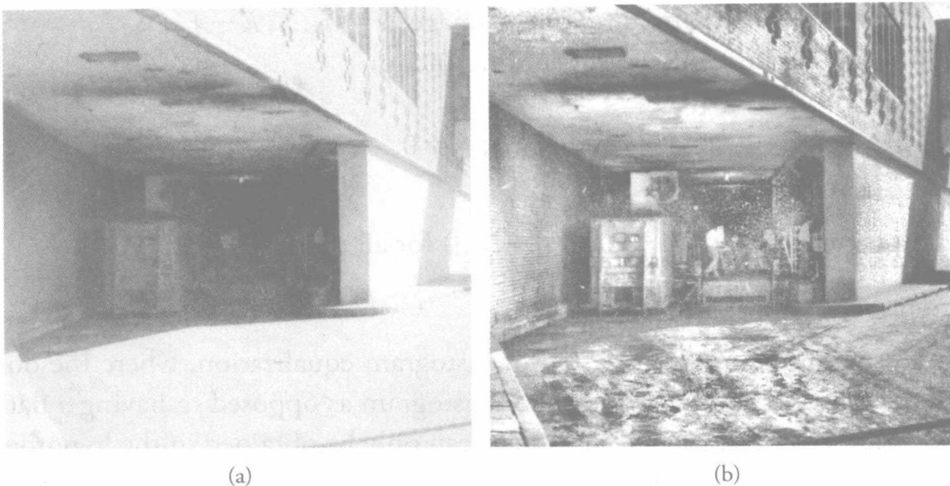
In the practical case of histogram shaping for images with quantized gray values, the desired CDF $Q(k)$ is discrete, and $Q^{-1}(k)$ must be defined carefully. In most cases, $Q(k)$ is either empirically stated or computed from another image (histogram matching problem). Then, $Q^{-1}(k)$ can be defined as

$$Q^{-1}(k) = \min_l \{l : Q(l) \geq k\} \tag{3.51}$$

which completes the specification of the histogram-shaping method.

**Local Contrast Manipulation by Pixel-Based Operators**

Contrast manipulation using global operators based on the histogram of the entire image does not produce satisfactory results if the input image has a bimodal



(a) (b)

**Figure 3.19** Adaptive histogram equalization: (a) input image containing both light and dark areas and (b) output image with $50 \times 50$ partially overlapping windows.

histogram. An example image with a bimodal histogram containing both very bright and very dark areas is shown in Figure 3.19(a). In such cases, we can achieve adaptive contrast enhancement by applying pixel-based contrast operators on local sliding image windows that are possibly overlapping, such that each window has a unimodal histogram containing either dark or bright areas.

### 3.4.2 Spatial Filtering for Tone Mapping and Image Sharpening

We begin this section with a tone-mapping method that is specifically developed for dynamic range compression of HDR images. We next discuss the retinex filter and unsharp masking and its extension based on bi-lateral filtering, which have been proposed as image-enhancement methods, but can also be used for dynamic range compression of HDR images.

**Digital Dodging-and-Burning for Tone Mapping [Rei 02]**

Reinhard [Rei 02] proposed a tone-reproduction operator for dynamic range compression of HDR images that is inspired by the photographic dodging-and-burning process. Dodging-and-burning is an artistic photographic printing technique where some light is manually withheld (dodging) from a region during development, or more light is added (burning) to a region. This will lighten or darken selected regions

in the final print relative to what it would be if the same amount of light were used for all regions of the print. The digital processing has two steps:

1. Luminance scaling for a given scene key value: Compute the geometric mean of the scene luminance:

$$\overline{L}_s = e^{\frac{1}{N}\sum_{x_1,x_2} \log(\delta + L_s(x_1,x_2))} \tag{3.52a}$$

where $L_s(x_1,x_2)$ is the scene luminance at pixel $(x_1,x_2)$, $N$ is the number of pixels in the image, and $\delta$ is a small constant to avoid singularity if black pixels are present in the image. Then, for a given key $a$, the luminance values are scaled as

$$L(x_1,x_2) = \frac{a}{\overline{L}_s} L_s(x_1,x_2) \tag{3.52b}$$

The key of a scene indicates whether it is subjectively light, normal, or dark. A predominantly white scene would be high-key, and a dim scene would be low-key. The value of $a$ is typically between 0.09 (low-key) and 0.36 or higher (high-key). For normal key scenes, $a = 0.18$.

2. Dynamic range compression: If the dynamic range of the image exceeds that of the display, as would be the case in HDR images, all luminance values cannot be displayed, and high values should be saturated by a compressive function. A simple compressive function is of the form

$$L_d(x_1,x_2) = \frac{L(x_1,x_2)}{1 + L(x_1,x_2)} \tag{3.52c}$$

where $L_d(x_1,x_2)$ denotes display values. However, this pixel-based scaling may result in loss of some important image details. A photographer would resort to dodging-and-burning to vary exposure locally to overcome this problem. A digital operator that resembles local dodging-and-burning process may be given by

$$L_d(x_1,x_2) = \frac{L(x_1,x_2)}{1 + V_1(x_1,x_2,s_m(x_1,x_2))} \tag{3.52d}$$

where $V_1(x_1,x_2,s_m(x_1,x_2))$ is a local average for pixel $(x_1, x_2)$ with a properly chosen neighborhood [Rei 02]. It is possible to compute the local average by means of a bi-lateral filter to avoid a halo effect around sharp edges.

### Unsharp Masking

Unsharp masking (USM) is a spatial-filtering operation that enhances medium to high frequencies, including edges. The USM filter is given by

$$g(n_1,n_2) = s_L(n_1,n_2) + \beta[s(n_1,n_2) - s_L(n_1,n_2)] \tag{3.53}$$

where $s_L(n_1, n_2)$ is low-pass filtered input image $s(n_1, n_2)$ and $\beta$ is the filter gain.

In the frequency domain, we have

$$G(e^{j\omega_1},e^{j\omega_2}) = S_L(e^{j\omega_1},e^{j\omega_2}) + \beta[S(e^{j\omega_1},e^{j\omega_2}) - S_L(e^{j\omega_1},e^{j\omega_2})] \tag{3.54}$$

The operation of the filter in the frequency domain is demonstrated by an example of a 1D signal in Figure 3.20.

The difference, $s(n_1, n_2) - s_L(n_1, n_2)$, is a detail image with medium-to-high frequency content since low frequencies are subtracted as shown in Figure 3.20. Hence, the basic idea of USM is to decompose an input image into a low-pass and a detail image. The detail image is multiplied by a gain factor $\beta$ and then added back to the low-pass filtered image. The result is an image with medium to high frequencies (texture and edges) boosted or enhanced.

The frequency range of the difference image determines what frequencies shall be enhanced in the output image, which is controlled by varying the bandwidth parameter of the low-pass filter. The low-pass filtered image can be computed in the

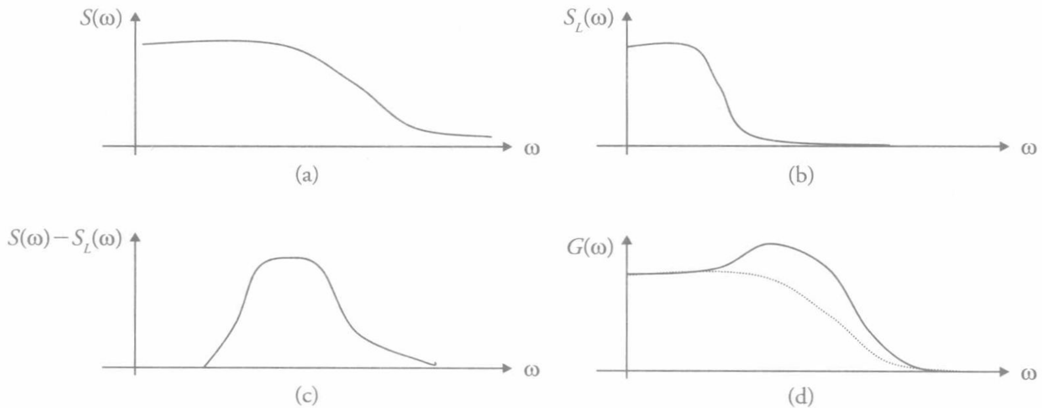



**Figure 3.20** Illustration of the concept of unsharp masking: (a) original signal spectrum; (b) low-pass filtered signal spectrum; (c) difference spectrum scaled by $\beta$; and (d) spectrum of the enhanced signal obtained by adding (b) and (c).

spatial domain or in the DFT domain using a separable 2D Gaussian or uniform (box filter) impulse response over a rectangular support. Fast implementation of box filtering [McD 81] was discussed in Section 3.1.1.

### Adaptive Filtering

The basic idea of unsharp masking has been extended to adaptive contrast manipulation as well as matching the dynamic range of an image to that of the display [Pel 82, Dur 02]. Peli and Lim [Pel 82] merged nonlinear pixel scaling and spatial filtering in an adaptive-filtering framework:

$$g(n_1,n_2) = T[s_L(n_1,n_2)] + \beta[s_L(n_1,n_2)][s(n_1,n_2) - s_L(n_1,n_2)] \tag{3.55}$$

where $T[\cdot]$ is a nonlinear point operation to match the dynamic range of the image to that of the display medium, and the filter gain $\beta$ changes according to the value of the smoothed image. The block diagram of the adaptive-filtering framework is depicted in Figure 3.21(a). Examples of $T[\cdot]$ and $\beta(s_L)$ are shown in Figure 3.21(b).

Durand and Dorsey [Dur 02] proposed replacing the simple Gaussian low-pass filter with a bi-lateral filter and used the resulting nonlinear filter for dynamic-range

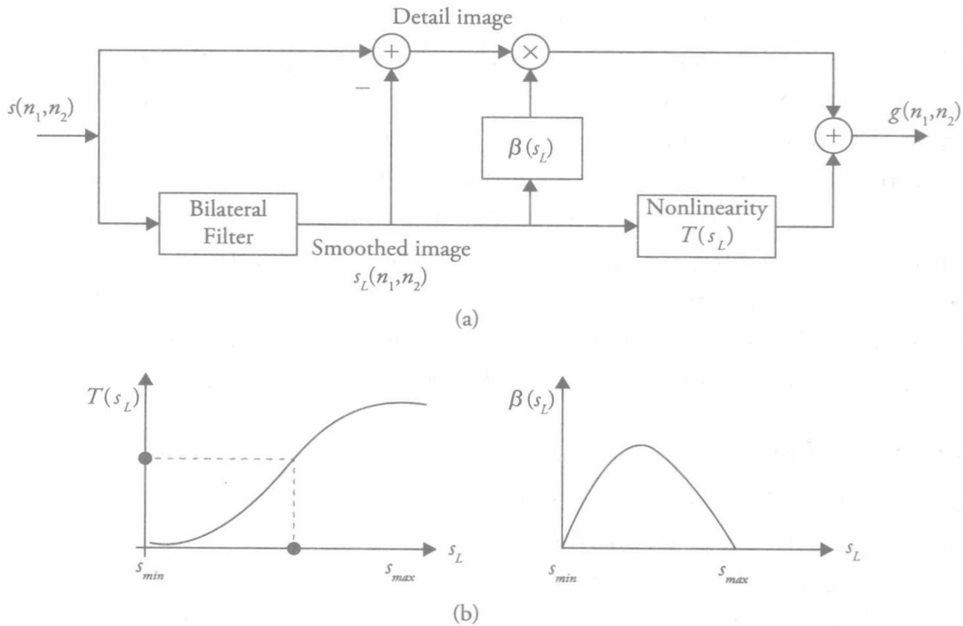



**Figure 3.21** Adaptive image enhancement: (a) block diagram of the adaptive filter and (b) examples of nonlinear functions $T[s_L(n_1, n_2)]$ and $\beta[s_L(n_1, n_2)]$.

compression of HDR images. They compress the dynamic range of only the smoothed luminance image (modeling scene illumination) by a choice of $T[s_L(n_1, n_2)]$ and preserve the detail image as well as the Cr and Cb components.

### Retinex

Low-contrast images are often a result of unfavorable illumination. We can obtain a better image by removing the effect of undesired illumination. Observed image luminance $L(x_1, x_2)$ can be modeled by the product of incident illumination $I(x_1, x_2)$ and scene reflectance $R(x_1, x_2)$, as

$$L(x_1, x_2) = I(x_1, x_2) R(x_1, x_2) \tag{3.56a}$$

Taking the logarithm of both sides, we transform the image to the log-luminance domain:

$$\begin{aligned} l(x_1, x_2) &= \log L(x_1, x_2) = \log I(x_1, x_2) + \log R(x_1, x_2) \\ &= i(x_1, x_2) + r(x_1, x_2) \end{aligned} \tag{3.56b}$$

where the effect of undesired illumination is additive. Retinex refers to a model and algorithm that was proposed by Land and McCann [McC 04] for removal of an undesired illumination component, which is modeled by an additive ramp (gradient) in the log-luminance domain.

There are various retinex algorithms that implement an iterative sequence of pixel comparisons at various scales or distances. Each iteration consists of so-called ratio, product, reset, and average operations. The overall effect of these iterative computations is low-pass filtering in the log-luminance domain. The McCann99 retinex implemented over a multi-resolution pyramid where the top-level is not larger than $5 \times 5$ pixels (MATLAB implementation available) [Fun 04] and the multi-scale retinex with color restoration [Rah 04] are the most popular implementations.

### Homomorphic Filtering

Related to retinex, filtering in the log-intensity domain (since human visual system perceives the logarithm of intensity) is known as homomorphic filtering [Opp 68]. The block diagram of homomorphic filtering is depicted in Figure 3.22, where a linear filter is used in the log-intensity domain. A high-pass filter may be employed to remove the low-frequency illumination component similar to retinex and obtain an image with more even illumination. Alternatively, a low-pass filter may be used

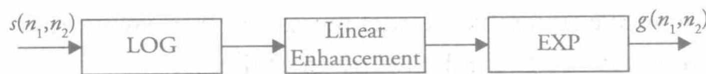



**Figure 3.22** Block diagram for homomorphic filtering.

to eliminate multiplicative intensity-domain noise. The main difference between retinex and homomorphic filtering is that in homomorphic filtering there is an exponentation after filtering that takes the output image back to the intensity domain for display.

## 3.5 Image Denoising

Image-capture mechanisms are not perfect. Images suffer from graininess due to photon noise, electronic noise, quantization noise, and impulsive noise due to sensor cell defects. Speckle noise is common in radar-image sequences and biomedical cine-ultrasound sequences. As a result, all images are contaminated by noise to some extent, which may or may not be visible. Even if noise may not be perceived at full-speed video due to the temporal-masking effect of the eye, it often leads to poor-quality "freeze-frame" still images. The signal-to-noise ratio (SNR) is an important imaging parameter, and it varies with the imaging modality and device.

Besides resulting in visually displeasing images and masking image details, noise also poses serious problems with solving ill-posed image processing problems. In image restoration and super-resolution, noise is the fundamental limitation in recovering high-frequency information. In motion estimation, it is important to distinguish intensity variations due to motion from those due to noise. In image/video compression, noise increases the entropy, hindering effective compression.

This section presents spatial denoising filters, where a single image is processed. Multi-frame video denoising is studied in Chapter 6. We can classify spatial noise filters in several ways: i) linear vs. nonlinear, ii) shift-invariant vs. adaptive, iii) local vs. non-local, and iv) pixel-wise vs. block-wise. Most filters fall into more than one of these classes, e.g., local, pixel-wise, adaptive or non-local, block-wise, nonlinear. After discussing image and noise modeling in Section 3.5.1, Section 3.5.2 introduces linear space-invariant filters that may be implemented in the spatial or transform domain. Local adaptive filters, such as the local LMMSE filters and directional filters, are covered in Section 3.5.3. We study nonlinear filters, such as order statistics filters, wavelet shrinkage, and bi-lateral filters in Section 3.5.4. Non-local filters, such as the non-local means (NLM) and BM3D, are introduced in Section 3.5.5.

A unifying data-adaptive (steerable) kernel regression framework that covers most of these filters and iterative filters can be found in [Mil 13].

### 3.5.1 Image and Noise Models

It is obvious that exact separation of fluctuations in image intensity due to noise from genuine image detail is impossible. All denoising methods are based on modeling self-similarity and sparsity characteristics of images and statistical characteristics of noise using models with varying complexity. Our ability to separate the signal from noise depends on how well various models allow such separation.

**Noise Models**

The noise can be modeled as additive or multiplicative, white or colored, and signal-dependent or signal-independent. A simple additive noise model is given by

$$y(n_1,n_2) = s(n_1,n_2) + v(n_1,n_2) \tag{3.57}$$

where $s(n_1, n_2)$ and $v(n_1, n_2)$ denote the ideal image and noise, respectively.

A noise source is called white noise, if all noise samples are uncorrelated with each other, i.e.,

$$\begin{aligned} E\,\{v(n_1,n_2)\ v(i_1,i_2)\} &= \sigma_v^2\delta(n_1-i_1,n_2-i_2) \\ &= \begin{cases} \sigma_v^2 & \text{if } (n_1,n_2)=(i_1,i_2) \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{3.58}$$

where $E\{\cdot\}$ denotes the expectation operator and $\sigma_v^2$ is the variance of the noise. Independent and identically distributed (i.i.d.) is a stronger condition than white.

A noise source is signal-independent if

$$E\,\{s(n_1,n_2)\ v(i_1,i_2)\} = 0 \text{ for all } (n_1,n_2) \text{ and } (i_1,i_2) \tag{3.59}$$

For example, photon noise and film grain are signal-dependent, whereas charge-coupled device (CCD) sensor noise and quantization noise are usually modeled as white, Gaussian, and signal-independent. Ghosts in TV images can also be modeled as signal-dependent noise. Other noise models can be found in Chapter 4.5 of [Bov 00].

The level of noise in an image is specified in terms of the signal-to-noise ratio (SNR), defined as

$$SNR = 10\log_{10}\frac{\sigma_s^2}{\sigma_v^2} \tag{3.60}$$

in decibels (dB), where $\sigma_s^2$ is the variance of the original noise-free image. If the SNR is below some level, typically 30 dB, i.e., the noise variance is more than 1/1000 of that of the image, the noise becomes visible as a pattern of graininess, resulting in a degradation of the image quality.

### *Estimation of the SNR*

In general, we need to estimate the variance $\sigma_v^2$ of the noise and variance $\sigma_s^2$ of the noise-free image from the given noisy image. This requires an ergodicity assumption, which allows us to estimate ensemble image statistics from a given sample image.

The variance of the noise can be estimated from a flat (untextured) image region, where the variance should ideally be zero in the absence of any noise. To this effect, we manually mark a flat rectangular region $W$ and first estimate its mean

$$\hat{\mu}_{y\in W} = \frac{1}{M}\sum_{(i_1,i_2)\in W} y[i_1,i_2] \tag{3.61}$$

where $M$ is the number of pixels in the selected region. The variance of the noise is estimated as

$$\hat{\sigma}_v^2 = \hat{\sigma}_{y\in W}^2 = \frac{1}{M}\sum_{(i_1,i_2)\in W}(y[i_1,i_2] - \hat{\mu}_{y\in W})^2 \tag{3.62}$$

In order to estimate the variance of the noise-free image, we repeat this procedure, this time over the whole picture to obtain $\hat{\sigma}_y^2$. Then, the estimate of $\sigma_s^2$ is given by

$$\hat{\sigma}_s^2 = \max\{\hat{\sigma}_y^2 - \hat{\sigma}_v^2, 0\} \tag{3.63}$$

so that $\hat{\sigma}_s^2$ is always non-negative.

### Image Models and Performance Limits

Various image models that have been used in image denoising are summarized in Appendix B. Minimizing a cost function based on the $l^2$-norm of the error (expressed as the minimum mean square error) subject to a global smoothness constraint (using

a homogeneous random field model) results in the well-known Wiener filter given by Eqn. (3.65). Minimizing the $l^1$-norm (expressed as the sum absolute error) yields the median filter. The wavelet shrinkage has been shown to minimize the $l^0$-norm of the estimation error, which is a sparseness measure. Non-local patch-similarity based models have also been used (see Section 3.5.5) for image denoising. Clearly, performance limits of denoising filters are strongly related to how well the image and noise models match the real situation. A lower bound for achievable mean-square error derived in [Cha 10] suggests that there may be room for further improvement to reach the performance limits in image denoising.

### 3.5.2 Linear Space-Invariant Filters in the DFT Domain

Linear space-invariant (LSI) denoising filters can be designed and analyzed using frequency-domain concepts, and are easier to implement. Natural images have more energy at low frequencies than at high frequencies, and the spectrum of white noise is flat as illustrated in Figure 3.23. An LSI noise reduction filter typically attenuates frequencies where the noise power exceeds the signal power. However, in LSI denoising, there is an inherent tradeoff between noise reduction and blurring of image detail. This is because any LSI denoising filter is essentially a low-pass filter, also called a smoothing filter (see Section 3.1.1), which suppresses high frequencies. As a result, high-frequency image content is also attenuated, causing blurring.

The linear minimum mean-square error (LMMSE) filter is the optimal LSI filter that yields the minimum mean-square error estimate of the ideal image; i.e., it is the optimal filter in the minimum mean-square error sense among all linear filters. Loosely speaking, it determines the best cutoff frequency for low-pass filtering given the power spectra of the ideal image and the noise. In the following, we derive both the infinite-impulse response (IIR) and finite-impulse response (FIR) LLMSE filters,

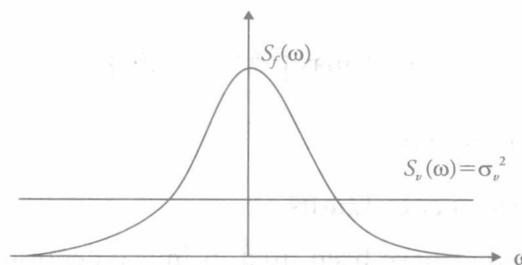



**Figure 3.23** Tradeoff between noise reduction and blurring in LSI denoising filters.

using the principle of orthogonality, assuming that image and noise are wide-sense stationary; i.e., their means are constant and their correlation functions are shift-invariant. The image and noise are assumed to be zero mean without loss of generality, since any non-zero mean can be removed prior to filtering.

**IIR Wiener Filter**

The input-output relationship for the IIR LMMSE filter can be expressed in the form of a convolution, given by

$$\hat{s}(n_1, n_2) = \sum_{i_1=-\infty}^{\infty} \sum_{i_2=-\infty}^{\infty} h(i_1, i_2)\ y(n_1 - i_1, n_2 - i_2) \tag{3.64}$$

where $y(n_1, n_2)$ is the noisy image, $\hat{s}(n_1, n_2)$ denotes the LMMSE estimate of the ideal image $s(n_1, n_2)$, and $h(n_1, n_2)$ is the impulse response of the filter.

The principle of orthogonality states that estimation error, $s(n_1, n_2) - \hat{s}(n_1, n_2)$, at each pixel should be orthogonal to every sample of the observed image, which can be expressed as

$$\langle s(n_1, n_2) - \hat{s}(n_1, n_2), y[k_1, k_2] \rangle = E\{(s(n_1, n_2) - \hat{s}(n_1, n_2))\, y[k_1, k_2]\} = 0 \tag{3.65}$$

for all $(n_1, n_2)$ and $(k_1, k_2)$, where the inner product $\langle \cdot, \cdot \rangle$ is defined in terms of the expectation operator $E\{\cdot\}$. Thus, orthogonality means uncorrelatedness.

Substituting (3.64) into (3.65), and simplifying the resulting expression, we obtain

$$E\left\{\sum_{i_1=-\infty}^{\infty} \sum_{i_2=-\infty}^{\infty} h(i_1, i_2)\ R_{yy}(n_1 - i_1 - k_1, n_2 - i_2 - k_2)\right\} = R_{sy}(n_1 - k_1, n_2 - k_2)$$
$$\text{for all } (n_1, n_2) \text{ and } (k_1, k_2) \tag{3.66a}$$

where

$$R_{yy}(n_1 - i_1 - k_1, n_2 - i_2 - k_2) = E\{y(n_1 - i_1, n_2 - i_2)\ y(k_1, k_2)\}$$

denotes the autocorrelation function of the observations, and

$$R_{sy}(n_1 - k_1, n_2 - k_2) = E\{s(n_1, n_2)\ y(k_1, k_2)\}$$

is the cross-correlation between the ideal image and the observed image. The double summation can be expressed as a 2D convolution:

$$h(n_1,n_2)**R_{yy}(n_1,n_2)=R_{sy}(n_1,n_2) \tag{3.66b}$$

which is called the discrete Wiener–Hopf equation. The expression (3.66) defines the impulse response of the noncausal, IIR Wiener filter, also known as the unrealizable Wiener filter. This filter is unrealizable because an infinite-time delay is required to compute an output sample.

We can obtain the frequency response of the unrealizable Wiener filter by taking the 2D Fourier transform of both sides of (3.66b), which results in

$$H(f_1,f_2)=\frac{P_{sy}(f_1,f_2)}{P_{yy}(f_1,f_2)} \tag{3.66c}$$

where $P_{yy}(f_1,f_2)$ and $P_{sy}(f_1,f_2)$ are the auto- and cross-power spectra, respectively. The derivation, up to this point, is quite general, in the sense that it only assumes that the filter (3.64) is linear, uses the principle of orthogonality (3.65), and is independent of the image and noise models. However, derivation of the expressions $P_{yy}(f_1,f_2)$ and $P_{sy}(f_1,f_2)$ requires a problem-specific observation model.

For the denoising problem, the auto- and cross-power spectra in (3.66) can be derived from the observation model (3.57) and the noise model (3.58) and (3.59) as

$$\begin{aligned}R_{sy}(n_1,n_2)&=E\{s(i_1,i_2)\ y(i_1-n_1,i_2-n_2)\}\\&=E\{s(i_1,i_2)\ s(i_1-n_1,i_2-n_2)\}+E\{s(i_1,i_2)v(i_1-n_1,i_2-n_2)\}\\&=R_{ss}(n_1,n_2)\end{aligned} \tag{3.67a}$$

and

$$\begin{aligned}R_{yy}(n_1,n_2)&=E\ \{(s(i_1,i_2)+v(i_1,i_2))(s(i_1-n_1,i_2-n_2)+v(i_1-n_1,i_2-n_2))\}\\&=R_{ss}(n_1,n_2)+R_{vv}(n_1,n_2)\end{aligned} \tag{3.67b}$$

where we assume that the image and noise are uncorrelated. Hence, the power spectra $P_{sy}(f_1,f_2)=P_{ss}(f_1,f_2)$ and $P_{yy}(f_1,f_2)=P_{ss}(f_1,f_2)+P_{vv}(f_1,f_2)$ can be obtained by

taking the 2D Fourier transform of the respective correlation functions. Then, the frequency response of the Wiener filter becomes

$$H(f_1, f_2) = \frac{P_{ss}(f_1, f_2)}{P_{ss}(f_1, f_2) + P_{vv}(f_1, f_2)} \tag{3.68}$$

We observe that $P_{vv}(f_1, f_2) = \sigma_v^2$ since the noise is white and the power spectrum of the noise-free image can be approximated with that of the noisy image as $\hat{P}_{ss}(f_1, f_2) = |Y(f_1, f_2)|^2$. It is well-known that the estimate can be improved by sectioning the image and averaging the spectrum estimates over the sections.

The Wiener filter is a low-pass filter, since the image power diminishes at high frequencies, which implies that the filter frequency response goes to zero at high frequencies, whereas at low frequencies the noise power is negligible compared to the image power so that the frequency response of the filter approaches one.

A realizable approximation to the filter (3.68) can be obtained by frequency-sampling design, where the filter frequency response $H(f_1, f_2)$ is sampled in the frequency domain using $N_1 \times N_2$ samples. This filter can be efficiently implemented using $N_1 \times N_2$ fast Fourier transform (FFT). The frequency-sampling design is equivalent to approximating the impulse response $h(n_1, n_2)$ of the IIR filter with an $N_1 \times N_2$ FIR filter with the impulse response $\tilde{h}(n_1, n_2)$, given by

$$\tilde{h}(n_1, n_2) = \sum_{r_1=-\infty}^{\infty} \sum_{r_2=-\infty}^{\infty} h(n_1 - N_1 r_1, n_2 - N_2 r_2) \tag{3.69}$$

Note that the frequency-sampling design method suffers from spatial-domain aliasing. In practice, this may be negligible, provided that $N_1$ and $N_2$ are reasonably large, e.g., $N_1 = N_2 = 512$ or larger.

## 3.5.3 Local Adaptive Filtering

Linear shift-invariant (LSI) filters limit our ability to separate genuine image detail from noise, because they are based on wide-sense stationary (homogeneous) image models. Local image models offer rich possibilities for adaptive image processing that overcomes limitations of LSI filters. Geman and Geman [Gem 84] model local interactions between pixels using Markov random field (MRF) models and use a nonlinear Bayesian framework for denoising and image restoration. MRF models and optimization methods are reviewed in Appendices C and D, respectively. In this section, we discuss two simple (non-iterative) filters: an adaptive LMMSE filter and a directional filter.

### FIR-LMMSE Filter

Alternatively, we can pose the denoising problem as an optimal linear shift-invariant FIR filter design problem. Assuming the observed image and the estimate are $N \times N$ arrays, the FIR-LMMSE filter can be expressed in vector-matrix form as

$$\hat{\mathbf{s}} = \mathbf{H}\mathbf{y} \tag{3.70}$$

where $\hat{\mathbf{s}}$ and $\mathbf{y}$ are $N^2 \times 1$ vectors formed by lexicographic ordering of the estimated and observed image pixels, and $\mathbf{H}$ is an $N^2 \times N^2$ matrix operator formed by coefficients of the FIR filter impulse response.

The principle of orthogonality can be stated in vector-matrix form as

$$E\{(\mathbf{s} - \hat{\mathbf{s}})\mathbf{y}^{\mathrm{T}}\} = \mathbf{0} \text{ (zero matrix)} \tag{3.71a}$$

which states that every element of $\mathbf{s} - \hat{\mathbf{s}}$ is uncorrelated with every element of $\mathbf{y}$. Substituting (3.70) into (3.71a), we obtain

$$E\{(\mathbf{s} - \mathbf{H}\mathbf{y})\,\mathbf{y}^{\mathrm{T}}\} = \mathbf{0}$$

which can be simplified as

$$E\{\mathbf{s}\,\mathbf{y}^{\mathrm{T}}\} = \mathbf{H}\; E\{\mathbf{y}\,\mathbf{y}^{\mathrm{T}}\} \tag{3.71b}$$

Then the FIR-LMMSE filter operator $\mathbf{H}$ can be obtained as

$$\mathbf{H} = \mathbf{R}_{sy}\mathbf{R}_{yy}^{-1} \tag{3.71c}$$

where $\mathbf{R}_{yy}$ is the auto-correlation matrix of the observed image and $\mathbf{R}_{sy}$ is the cross-correlation matrix of the ideal and observed images.

Given the observation model, we can easily show, as in the derivation of the IIR filter, that $\mathbf{R}_{sy} = \mathbf{R}_{ss}$, and $\mathbf{R}_{yy} = \mathbf{R}_{ss} + \mathbf{R}_{vv}$, where $\mathbf{R}_{ss}$ and $\mathbf{R}_{vv}$ are the auto-correlation matrices of the ideal image and the observation noise, respectively. Then the filter operator becomes

$$\mathbf{H} = \mathbf{R}_{ss}\left[\mathbf{R}_{ss} + \mathbf{R}_{vv}\right]^{-1} \tag{3.72}$$

Observe that the implementation of the filter (3.72) requires inversion of an $N^2 \times N^2$ matrix. For a typical digital image, e.g., $N$=512 or larger, this is a formidable task.

However, assuming that the image and noise are wide-sense stationary, i.e., they have constant mean vectors (taken as zero without loss of generality) and spatially invariant correlation matrices, the matrices $\mathbf{R}_{ss}$ and $\mathbf{R}_{vv}$ are block-Toeplitz. It is common to approximate block-Toeplitz matrices by block-circulant ones, which can be diagonalized through the 2D-DFT operation [Gon 07]. The reader can readily see that the resulting frequency-domain FIR filter expression is identical to that obtained by sampling the frequencies $(f_1, f_2)$ in Eqn. (3.68).

**Adaptive LMMSE Filter**

As a compromise between the complexity of (3.72) and preserving important image detail, a simple space-varying image model, where the local image characteristics are captured in a space-varying mean, has been proposed [Lee 80, Kua 85]. The residual image after removing the local mean is modeled by a white Gaussian process with a space-varying variance. This section presents a spatially adaptive LMMSE estimator that is based on this model. The resulting filter is easy to implement, yet avoids blurring in the vicinity of edges and other detail.

We define a residual image as

$$r_s(n_1,n_2) = s(n_1,n_2) - \mu_s(n_1,n_2) \tag{3.73}$$

where $\mu_s(n_1, n_2)$ is a spatially varying mean image. The residual image $r_s(n_1, n_2)$ is modeled by a white process, i.e., its correlation matrix, given by

$$R_{rr}(n_1,n_2) = \sigma_s^2(n_1,n_2)\ \delta(n_1,n_2)$$

is diagonal. Note that the variance, $\sigma_s^2(n_1, n_2)$, of the residual also varies from pixel to pixel. It follows that the residual of the observed image, defined as

$$r_y(n_1,n_2) = y(n_1,n_2) - \mu_y(n_1,n_2) \tag{3.74}$$

where $\mu_y(n_1, n_2)$ is the local mean of the observed image, is zero-mean and white, because the noise is assumed to be zero-mean and white. Furthermore, from (3.57),

$$\mu_y(n_1, n_2) = \mu_s(n_1, n_2)$$

since the noise is zero-mean.

Applying the FIR-LMMSE filter (3.70) to the residual observation vector, $\mathbf{r}_y = \mathbf{y} - \boldsymbol{\mu}_y$, which is a zero-mean, wide-sense stationary image, we have

$$\hat{\mathbf{r}}_s - \boldsymbol{\mu}_s = \mathbf{H}\,\mathbf{r}_y = \mathbf{R}_{rr}\left[\mathbf{R}_{rr} + \mathbf{R}_{vv}\right]^{-1}(\mathbf{y} - \boldsymbol{\mu}_y) \tag{3.75a}$$

The matrix $\mathbf{R}_{rr}$ is diagonal because $r_s(n_1, n_2)$ is white. Then, the above vector-matrix expression simplifies to the scalar form

$$\hat{s}(n_1,n_2) = \mu_y(n_1,n_2) + \frac{\sigma_s^2(n_1,n_2)}{\sigma_s^2(n_1,n_2)+\sigma_v^2}[y(n_1,n_2) - \mu_y(n_1,n_2)] \tag{3.75b}$$

which has a predictor-corrector structure and $\sigma_s^2(n_1, n_2) / (\sigma_s^2(n_1, n_2) + \sigma_v^2)$ is called the filter gain.

The adaptive LMMSE filter (3.75) requires estimation of the local mean $\mu_y(n_1, n_2)$ and the local variance $\sigma_s^2(n_1, n_2)$ at each pixel, which can be computed over an $M \times M$ local window $W_{n_1,n_2}$ centered at the pixel $(n_1, n_2)$ as

$$\mu_s(n_1,n_2) = \mu_y(n_1,n_2) = \frac{1}{M^2}\sum_{(i_1,i_2)\in W_{n_1,n_2}} y(i_1,i_2) \tag{3.76}$$

since the noise has zero-mean, and

$$\sigma_y^2(n_1,n_2) = \frac{1}{M^2}\sum_{(i_1,i_2)\in W_{n_1,n_2}} \left(y(i_1,i_2) - \mu_y(n_1,n_2)\right)^2 \tag{3.77}$$

In order to avoid a negative variance estimate, we have

$$\hat{\sigma}_s^2(n_1,n_2) = \max\{\sigma_y^2(n_1,n_2) - \sigma_v^2, 0\} \tag{3.78}$$

Note that when $\hat{\sigma}_s^2$ is small, indicating a uniform image region, the filter gain is negligible, and the adaptive LMMSE filter approaches a direct averaging filter. On the other hand, when $\hat{\sigma}_s^2$ is large compared to $\sigma_v^2$, which indicates the presence of edges or high-contrast texture, the filter gain approaches one, and edges/texture are preserved by effectively turning the filter off. Consequently, some noise is left around the edges, which may not be visible due to the masking effect of human vision. However, this may be visually disturbing in low SNR cases.

**Directional Filtering**

An alternative approach for edge-preserving filtering is directional filtering, where we filter along the edges, but not across them. The directional filtering approach may be superior to adaptive LMMSE filtering in low SNR cases, since noise around edges can effectively be eliminated by filtering along the edges, as opposed to turning the filter off in the neighborhood of edges.

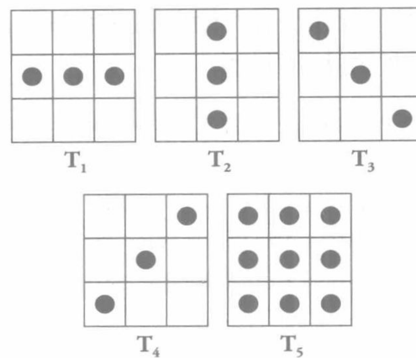



**Figure 3.24** Directional filtering kernels.

In directional filtering, possible edge orientations are typically quantized into four angles, 0°, 45°, 90°, and 135°, and five FIR filter kernels, one for each orientation and one for non-edge regions, are defined. The supports of the edge-oriented FIR filters are depicted in Figure 3.24. In general, there exist two approaches for directional filtering: i) select the most uniform support out of the five at each pixel according to a criterion of uniformity or by edge detection [Dav 75], or ii) apply an edge-adaptive filter within each kernel at each pixel and cascade the results [Cha 85].

1. *Method I: Kernel selection.* If the variance of the pixels within the non-edge kernel $\mathbf{T}_5$ is more than a pre-determined threshold, we decide that an edge is present at that pixel. Then, one of the edge kernels $\mathbf{T}_1 - \mathbf{T}_4$ with the lowest variance (provided that it is lower than that of $\mathbf{T}_5$) is selected as the most likely edge orientation at that pixel. Filtering is performed by averaging pixels indicated by black dots in the respective kernel or by edge-oriented Gaussian filters, called anisotropic diffusion filters. Filtering along the edge direction avoids spatial blurring to a large extent.
2. *Method II: Cascade.* We use a spatially adaptive filter, such as the local LMMSE filter, within each of the five supports at each pixel. Recall that the local LMMSE filter is effectively off within those supports with a high variance, and approaches direct averaging as the signal variance goes to zero. Thus, when cascading five filters as $\mathbf{T} = \mathbf{T}_1\mathbf{T}_2\mathbf{T}_3\mathbf{T}_4\mathbf{T}_5$, where $\mathbf{T}_i$ is the local LMMSE filter applied over the respective kernel [Cha 85], effective filtering is performed only over those kernels with a small variance. This method avoids the support-selection problem.

Directional filtering, using either method, usually offers satisfactory noise reduction around edges, since at least one of the filters shall be active at every pixel.

### 3.5.4 Nonlinear Filtering: Order-Statistics, Wavelet Shrinkage, and Bi-Lateral Filtering

Nonlinear filtering refers to a very large family of image-processing operations; namely, any operation that is not linear. Nonlinear filters can be classified as pixel-wise or block-wise filters. Here, we focus on only a few popular classes of nonlinear denoising techniques, which are pixel-wise median/order-statistics filters and bi-lateral filters, and block-wise wavelet shrinkage filters.

#### Median Filtering

Median filtering is a nonlinear operation that is implicitly edge adaptive. The output of the median filter is given by the median of pixels within the support of the filter, expressed as

$$\hat{s}(n_1, n_2) = \text{Med}\{\, y(i_1, i_2)\} \quad \text{for} \quad (i_1, i_2) \in \boldsymbol{B}_{(n_1, n_2)} \tag{3.79}$$

where $\boldsymbol{B}_{(n_1,n_2)}$ denotes the filter support, e.g., a local neighborhood of pixel $(n_1, n_2)$, and "Med" denotes the median operation. For example, in $3 \times 3$ median filtering, there are nine pixels in the $3 \times 3$ neighborhood of a pixel, and the output for the center pixel is given by the intensity of the fifth ranked (by intensity value) pixel. Larger filters may be needed to remove large clusters of impulses. The median filter is edge-preserving since it rejects outliers, avoiding blurring across edges [Ata 80, Arc 91, Yin 96]. Fast algorithms for 2D (separable) median filtering exist [Ata 80].

#### Example: Median Filtering

We demonstrate how median filtering preserves edges by means of a simple example of 1D filtering. The original signal, in the form of two step edges, is randomly contaminated by impulse noise added to the shaded samples. We observe that the $3 \times 1$ mean filter spreads the noise to neighboring samples, while the $3 \times 1$ median filter effectively removes all noise in this example.

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Original signal | 10 | 10 | 10 | 10 | 10 | 10 | 40 | 40 | 40 | 40 | 40 | 80 | 80 | 80 | 80 | 80 | 80 |
| Noisy signal | 10 | 10 | 10 | 60 | 10 | 10 | 40 | 80 | 40 | 10 | 80 | 40 | 80 | 80 | 10 | 80 | 80 |
| Mean filtered | | 10 | 27 | 27 | 27 | 20 | 43 | 53 | 43 | 43 | 43 | 67 | 67 | 57 | 57 | 57 | |
| Median filtered | | 10 | 10 | 10 | 10 | 10 | 40 | 40 | 40 | 40 | 40 | 80 | 80 | 80 | 80 | 80 | |

### *Weighted Median Filtering*

In median filtering each sample in the filter support is given an equal emphasis. However, in some cases, median operation results in distortion around corners. The weighted median filter is an extension of the median filter where each sample $(i_1, i_2)$ is assigned a weight $w_{(i_1,i_2)}$. The weighting is achieved by replicating the $(i_1, i_2)th$ sample $w_{(i_1,i_2)}$ times. Then, the output of the weighted median filter is given by

$$\hat{s}(n_1,n_2) = \text{Med}\{w_{(i_1,i_2)} \diamond y(i_1,i_2)\} \quad \text{for} \quad (i_1,i_2) \in \boldsymbol{B}_{(n_1,n_2)} \tag{3.80}$$

where $\diamond$ is the replication operator. The properties of the filter vary depending on how weights are assigned. The reader is referred to [Yin 96] for further details.

We can compare the properties of the median filter with those of the mean filter:

1. The median of an odd number of samples $N_W$ is the sample with the smallest sum of absolute differences with other samples in a given set of samples. The sample mean is an estimate with the smallest square distance with all samples. Thus, sample median and sample mean provide an estimate $\beta$ that minimizes the criterion $D(\beta) = \sum_{i=1}^{N_W} | y_{(i)} - \beta |^p$ for $p = 1$ and $p = 2$, respectively.
2. The sample mean is the maximum likelihood (ML) estimate in the presence of Gaussian noise, whereas the sample median is the ML estimate in the presence of Laplacian noise, which has heavier tails than Gaussian, i.e., impulsive noise.

### Order-Statistics Filters

Order-statistics filters require rank ordering (sorting) pixel values in a neighborhood of the current pixel. An alpha-trimmed mean filter is an order-statistics filter that combines rank ordering and averaging operations to compute the output. In particular, $L$ smallest and $L$ largest pixel values in a local neighborhood are eliminated, and the average of remaining middle ranked samples are computed as

$$\hat{s}(n_1,n_2) = \frac{1}{N_W - 2L} \sum_{i=L+1}^{N_W - L} y_{(i)} \tag{3.81}$$

The alpha-trimmed mean filter becomes a mean filter for $L = 0$, and it approaches the median filter as $L \to N_W/2$. Therefore, it can be effective in suppressing a combination of Gaussian noise and impulsive salt-and-pepper noise by proper selection

of $L$. The median and order statistics filters can be made adaptive by adjusting the size of the local neighborhood to match local image and noise statistics. The reader is referred to [Arc 91] for details about multi-stage order statistic filters.

**Wavelet Shrinkage – Denoising Using Sparse Representations**

Natural images can be represented by a sparse vector in some transform domain, e.g., an orthogonal wavelet transform domain. Under sparse image modeling, additive noise leads to very low SNR on many transform coefficients with small magnitudes. A thresholding can simply detect and remove these coefficients resulting in robust and high-performance denoising. Denoising by wavelet shrinkage, proposed by Donoho [Don 95a], refers to hard or soft thresholding of orthogonal wavelet transform coefficients, where it is assumed that larger coefficients represent "signal" and small coefficients are "noise." All wavelet shrinkage methods consist of the following three main steps: i) linear forward wavelet transform, ii) nonlinear shrinkage, and iii) inverse wavelet transform. Many different wavelet shrinkage methods vary in the details of wavelet transform implementation and choice of shrinkage function and threshold value.

The selection of wavelet basis functions (analysis and synthesis filters), the number of resolution levels, and image-boundary handling in filtering all affect denoising performance. The nearly symmetric orthogonal wavelets are generally preferred for denoising, since orthogonal transform of white noise is white in the wavelet domain, and orthogonal transforms preserve the mean-squared error. Typical choices are 3 or 4 resolution levels using Symlet8 wavelets and periodic or symmetric boundary extension [Fod 03].

The shrinkage functions can be classified according to i) whether they use hard or soft thresholding, where hard thresholding of wavelet coefficient $w$ is defined by

$$\delta^H(w) = \begin{cases} w & \text{if } |w| > \lambda \\ 0 & \text{otherwise} \end{cases} \tag{3.82}$$

and soft thresholding is defined by

$$\delta^S(w) = \begin{cases} \operatorname{sgn}(w)(|w| - \lambda) & \text{if } |w| > \lambda \\ 0 & \text{otherwise} \end{cases} \tag{3.83}$$

and ii) whether they use a universal threshold or adaptive thresholds for different resolution levels or subbands. They also vary according to the criterion used for determination of the threshold value $\lambda$. Several different criteria exist to estimate the

best threshold, such as Stein's unbiased risk estimate (SURE), minimax, and Bayesian criteria [Fod 03, Lui 07].

VisuShrink uses a universal threshold $\lambda = \sigma_v \sqrt{2 \log N}$, where $\sigma_v^2$ is the variance of noise and $N$ is the number of wavelet coefficients. SureShrink selects an adaptive threshold for each sub-band to optimize the SURE criterion [Don 95b]. It combines universal threshold selection and scale-dependent adaptive threshold selection according to sparsity of subbands. The soft thresholding function is continuous; however, its first derivative is not continuous. Hence, Donoho's method searches for the optimal threshold within a finite set. SureShink can be summarized as: Assuming the wavelet coefficients are normalized by an estimate of $\sigma_v$, let $\mathbf{w}_j$ denote the vector formed by the normalized wavelet coefficients $w_n/\sigma_v$, $n = 1, \ldots, N_j$ in sub-band $j$ and $N_j$ be the number of wavelet coefficients in subband $j$. For each sub-band $j$, use a fixed universal threshold, given by

$$\lambda_j^U = \sqrt{2 \log N_j} \text{ if } \frac{1}{N_j} \sum_{n=1}^{N_j} \left( \left( \frac{w_n}{\sigma_v} \right)^2 - 1 \right) \le \frac{(\log_2 N_j)^{\frac{3}{2}}}{\sqrt{N_j}}$$

i.e., if only few wavelet coefficients are non-zero. Otherwise, use the SURE threshold, given by

$$\lambda_j^S = \arg\min_{\lambda > 0} \text{SURE}(\lambda, \mathbf{w}_j) \tag{3.84}$$

where

$$\text{SURE}(\lambda, \mathbf{w}_j) = N_j - 2M_j + \sum_{n=1}^{N_j} \left[ \min\left( \left| \frac{w_n}{\sigma_v} \right|, \lambda \right) \right]^2$$

and $M_j$ is the number of coefficients in sub-band $j$ whose absolute value is less than $\lambda$. Then, the denoised coefficients are given by $\hat{w}_n = \sigma_v \delta^S(w_n/\sigma_v)$.

BayesShrink is another scale-adaptive threshold estimation method that minimizes the Bayes risk [Cha 00]. The threshold is given by $\lambda = \sigma_v^2/\sigma_s$ where $\sigma_s^2$ is the variance of the noise-free signal (for each subband), which must be estimated from the noisy image.

### Bi-Lateral Filter

Bi-lateral filters perform combined domain and range filtering by some weighting of geometric proximity and similarity in the intensity or the CIE-Lab color space as introduced in Section 3.1.2. Bi-lateral filtering has been applied to image denoising

by proper choice of the kernel size $N \times N$, domain parameter $\sigma_p^2$, and range parameter $\sigma_q^2$ to match the noise statistics. The larger the range parameter, the closer the filter approaches Gaussian filtering. Hence, there is tradeoff between the amount of noise reduction and blurring of edges. In the presence of salt-and-pepper noise, a median filter may be applied prior to bi-lateral filtering. In bi-lateral filtering of color images, instead of the common practice of processing the luminance only, a perceptual color similarity metric in the CIE-Lab color space can be employed to smooth colors without color bleeding and blurring. Range filtering in the CIE-Lab color space is a natural way of processing color images, where only perceptually similar colors are averaged, and perceptually important edges are preserved.

### 3.5.5 Non-Local Filtering: NL-Means and BM3D

Most image-denoising methods exploit correlations between pixel intensities within a local neighborhood of a pixel, i.e., they assume pixel intensities within nearby locations are similar to each other, while noise samples are uncorrelated. However, this assumption breaks down near edges or texture regions, where contrast and/or color of pixels change suddenly. As a result, modeling similarity by geometric proximity of pixel locations (in the domain of the image) causes blurring and/or color bleeding artifacts in image denoising. To overcome this problem, bi-lateral filtering combines domain and range filtering (see Section 3.5.4). Alternatively, non-local filtering methods exploit non-local self-similarities in an image, i.e., range (intensity) similarity over non-local image areas. We study non-local means filtering (a patch-based range filtering method), and BM3D (a patch-based processing method in the transform domain) in more detail in the following.

**Non-Local Means Filtering**

The non-local means (NLM) filter locates patches, defined as fixed-size small windows, which can be overlapping or non-overlapping, that are similar to a patch centered at the current pixel, and denoise the current pixel by a weighted average of the center pixels of these similar patches. The main differences between bi-lateral and NLM filters are that the NLM filter replaces the single-pixel intensity similarity measure in bi-lateral filtering with a fixed size patch-based intensity similarity measure, and ignores the geometric proximity measure to exploit non-local self-similarities. Buades et al. [Bua 05] show that NLM filters are very effective for denoising.

We describe pixel-wise implementation of the NLM filter: Given a noisy image $y(\mathbf{n})$, the filtered intensity value $\hat{s}(\mathbf{n})$ for each pixel $\mathbf{n}$ can be computed as a weighted average of all similar pixels $\mathbf{k}$ in the image, given by

$$\hat{s}(\mathbf{n}) = \frac{1}{C(\mathbf{n})} \sum_{\mathbf{k} \in N(n)} w(\mathbf{n},\ \mathbf{k})\ y(\mathbf{k}) \tag{3.85}$$

where $C(\mathbf{n}) = \sum_{\mathbf{k}} \mathrm{w}(\mathbf{n}, \mathbf{k})$ is a normalizing constant and $N(\mathbf{n})$ denotes a $(2q+1) \times (2q+1)$ window centered at pixel $\mathbf{n}$ to search for similar patches. The weights $w(\mathbf{n}, \mathbf{k})$ depend on the sum of the absolute value of pixel-by-pixel similarity comparison of $(2r+1) \times (2r+1)$ patches $P_r$ centered at pixels $\mathbf{n}$ and $\mathbf{k}$, respectively, given by

$$d^2(\mathbf{n}, \mathbf{k}) = \frac{1}{(2r+1)^2} \sum_{\mathrm{i} \in P_r} [y(\mathbf{n} - \mathrm{i}) - y(\mathbf{k} - \mathbf{i})]^2 \tag{3.86}$$

The patch size is typically $5 \times 5$ ($r = 2$), except maybe in very noisy images. The search window has a limited range for computational efficiency reasons. The size of the search window typically varies between $21 \times 21$ ($q = 10$ for moderate noise) and $35 \times 35$ ($q = 17$ for large noise). The larger the number of similar pixels, the better the noise reduction. In order not to exclude the current pixel from its own estimate, the distance between the window centered at the current pixel and itself, $d(\mathbf{n},\mathbf{n})$ is set equal to the minimum of all other distances. The intensity similarity between two patches can also be weighted by a Gaussian kernel such that pixels closer to the center should have more weight in patch comparison.

The weights are designed to allow averaging center pixels of similar patches, which differ up to noise level with equal emphasis. If the noise samples are i.i.d. with zero mean and variance $\sigma^2$, the maximum distance between two identical patches due to noise can be $2\sigma^2$. That is, weights for patches with square distances smaller than $2\sigma^2$ are set to 1, while weights for patches with larger distances decrease rapidly according to the exponential rule

$$\mathrm{w}(\mathbf{n}, \mathbf{k}) = e^{-\frac{\max\{d^2(\mathbf{n}) - 2\sigma^2, 0\}}{h^2}} \tag{3.87}$$

where the parameter $h$ controls the decay of the weights as a function of intensity dissimilarity. If no matching patches are found for a pixel to satisfy $d^2(\mathbf{n}) < 2\sigma^2$, the filter may not alter the value of the pixel leaving the noise unprocessed.

**Block-Matching 3D (BM3D) Filtering**

Katkovnik et al. [Kat 10] classify noise filters as local/non-local and pixel-wise vs. block-wise (multi-point) filters. While the NL-means filter is an example of non-local, pixel-wise filtering, the BM3D is a non-local, block-wise filtering method, where possibly overlapping similar 2D-image blocks are grouped into 3D arrays.

BM3D applies collaborative filtering, which is a 3D-transform domain filtering, to such 3D arrays (groups). It is a block-wise filter, as opposed to a pixel-wise filter, since it outputs a group of processed 2D-image blocks, which are further aggregated to form the final image estimate. More specifically, BM3D consists of three main steps:

1. *Formation of groups:* Given the current block $N_\mathbf{n}$, similar blocks $N_\mathbf{k}$ are selected and stacked together as a 3D array. If all blocks contain the same structure except for the noise, then averaging pixels in the same location over all blocks would be the optimal estimator. In the more realistic case, there are some minor structural differences between the blocks that require collaborative filtering in the transform domain.
2. *Colloborative filtering:* The filter consists of 3D transformation of 3D groups, shrinkage of transform coefficients, and inverse 3D transformation. Similarity within the 3D groups of blocks implies that the resulting transform will be sparse. Shrinkage of transform coefficients attenuates noise while keeping the fine details shared by the group of blocks. Furthermore, significant improvements can be obtained by collaborative Wiener filtering [Dab 07]. The result is a 3D estimate that consists of a jointly filtered group of image blocks.
3. *Aggregation:* The filtered 2D blocks are returned to their original positions. Because these blocks overlap, we obtain multiple estimates for each pixel, which are combined by a weighted averaging procedure to form the final denoised estimate for each pixel.

BM3D has been shown to achieve state-of-the-art denoising performance in terms of both peak signal-to-noise ratio and subjective visual quality [Dab 07]. Extensions of BM3D to shape adaptive transform domain filtering and PCA on these adaptive-shape neighborhoods exist [Kat 10].

## 3.6 Image Restoration

Image restoration refers to deblurring of images degraded by optical smearing, including linear motion, out-of-focus imaging, or camera shake. Optical blurring arises when a single point object spreads over several image pixels, which may be due to relative motion between the object and camera or out-of-focus imaging. The extent of the spatial spread is determined by the point-spread function (PSF), which is the impulse response of the imaging system. In the discrete spatio-temporal domain, the general restoration problem can be formulated as solving a (possibly

underdetermined) set of simultaneous linear equations, which simplifies to a deconvolution problem in the case of spatially invariant blurs.

## 3.6.1 Blur Models

This section introduces the concept of a point spread function and discusses shift-invariant and shift-varying modeling of spatial blurring by the convolution and superposition summation, respectively. A vector-matrix model is also presented. Blurring in the temporal direction is often negligible, and will not be considered.

### Point-Spread Function

Modeling a point source (object) by a 2D impulse, the PSF defines the impulse response of an imaging system. We present PSF models for the most common sources of blur: out-of-focus blur, motion blur, and camera shake.

#### *Out-of-Focus Blur*

An ideal camera would form a point image at the focal plane of the camera for a point source. However, if the camera is out-of-focus, i.e., if the image plane (sensor) is moved away from the focal plane, then we observe a circle (with finite radius) image for a point source as depicted in Figure 3.25. This circle is commonly known as the circle of confusion.

Thus, we model the PSF of an out-of-focus imaging system with a uniform circle, called the circle of confusion, given by

$$h(x_1, x_2) = \begin{cases} \dfrac{1}{\pi R^2} & x_1^2 + x_2^2 \leq R^2 \\ 0 & \text{otherwise} \end{cases} \tag{3.88}$$

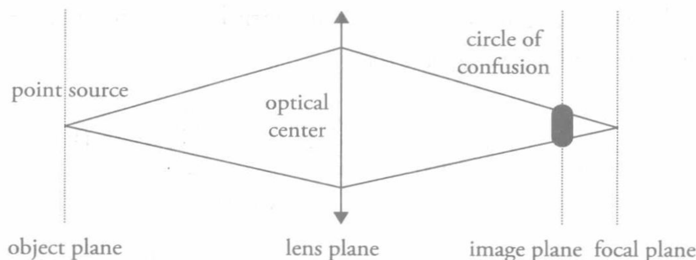



**Figure 3.25** Illustration of out-of-focus blur formation and circle of confusion.

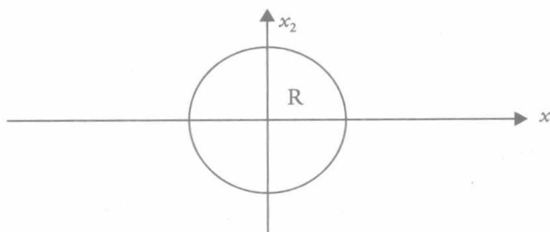



**Figure 3.26** PSF of out-of-focus blur.

where the radius of the circle, depicted in Figure 3.26, indicates the amount of out-of-focus. The discrete PSF $h(n_1,n_2)$ is obtained by sampling of $h(x_1,x_2)$ by means of integrating the area under the continuous PSF within each pixel element.

The frequency response of the out-of-focus blur is given by the Fourier transform of $h(x_1,x_2)$ as

$$H(u_1,u_2) = \frac{2\pi R}{\sqrt{u_1^2+u_2^2}} J_1\left(R\sqrt{u_1^2+u_2^2}\right) \tag{3.89}$$

where $J_1(\cdot)$ stands for the Bessel function of the first kind and order one, which has regular zero-crossings. Note that $H(u_1,u_2)$ is circularly symmetric.

### *Linear-Motion Blur*

Suppose there is relative translation between the camera and the scene at a constant velocity $\nu$ along direction $\theta$ with the horizontal axis of the image plane during the exposure interval $[0,t_e]$. Let's define the extent of the motion as $A=\nu t_e$. Then, the PSF of linear motion blur is given by

$$h(x_1,x_2) = \begin{cases} \dfrac{1}{A} & \text{if } \sqrt{x_1^2+x_2^2} \leq \dfrac{A}{2},\ \dfrac{x_1}{x_2} = -\tan\theta \\ 0 & \text{otherwise} \end{cases} \tag{3.90a}$$

In the special case, where the motion is parallel to the horizontal axis in the image plane, we have a 1D PSF given by

$$h(x_1,x_2) = \begin{cases} \dfrac{1}{A} & \text{if } -\dfrac{A}{2} \leq x_1 \leq \dfrac{A}{2},\ x_2 = 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.90b}$$

The frequency response of the 1D horizontal uniform motion blur can be expressed as

$$H(u_1,u_2)=\frac{\sin\left(\pi\frac{A}{2}u_1\right)}{\pi\frac{A}{2}u_1} \tag{3.91}$$

which has periodic zero-crossings. Discretization of the PSF is again accomplished by integrating the area under $h(x_1, x_2)$ over each pixel element. Motion blur becomes visible in scenes with fast action, such as in freeze frames of sports video.

Camera shake is another common source of image blur that can be modeled by uniform motion provided that camera rotation is negligible [Fer 06].

### Space-Invariant Spatial Blurring

If we have a linear space-invariant (LSI) blur, the blurred image can be modeled as the output of a linear filter whose impulse response is not a function of position within the image. This means the image is blurred exactly the same way at each pixel position, i.e., there is no significant parallax and image-plane rotation of the camera is small. A spatially invariant blur can be modeled by 2D convolution of an image with the spatial PSF $h(n_1, n_2)$ given by

$$y(n_1,n_2)=h(n_1,n_2)**s(n_1,n_2)+v(n_1,n_2) \tag{3.92a}$$

where $y(n_1, n_2)$, $s(n_1, n_2)$, and $v(n_1, n_2)$ denote the degraded image, ideal image, and noise, respectively. We assume that

1. $h(n_1, n_2)$ is real and non-negative for all $(n_1, n_2)$ due to physics of image formation, and
2. $\sum_{n_1}\sum_{n_2} h(n_1,n_2)=1$; i.e., no energy is gained or lost due to image blurring.

The model (3.92a) can be expressed in vector-matrix form as

$$\mathbf{y}=\mathbf{H}\,\mathbf{s}+\mathbf{v} \tag{3.92b}$$

where **y**, **s**, and **v** denote the observed image, the original image, and the noise that are lexicographically ordered into $N^2 \times 1$ vectors (for an $N \times N$ frame), respectively, and the matrix **H** characterizes the blur PSF in operator form. Note that for a spatially shift-invariant blur, **H** is a block-Toeplitz matrix.

### Shift-Varying Spatial Blurring

Image formation in the presence of a linear shift-varying (LSV) blur can be modeled as the output of a 2D-LSV system, denoted by **L**. We define the space-varying PSF of this imaging system as

$$h(n_1, n_2; i_1, i_2) = \mathbf{L}\{\delta(n_1 - i_1, n_2 - i_2)\} \tag{3.93}$$

which represents the image of a point source located at $(i_1, i_2)$, and may have infinite extent. It is well-known that a discrete image can be represented as a weighted and shifted sum of point sources as

$$s(n_1, n_2) = \sum\sum_{(i_1, i_2)\in S} s(i_1, i_2)\ \delta(n_1 - i_1, n_2 - i_2) \tag{3.94}$$

where $s(i_1, i_2)$ denotes the samples of the image and $\boldsymbol{S}$ denotes the image support. Then, applying the system operator to both sides of (3.94), using (3.93), and including the observation noise, $v(n_1, n_2)$, the blurred frame can be expressed as

$$y(n_1, n_2) = \sum\sum_{(i_1, i_2)\in S} s(i_1, i_2)\ h(n_1, n_2; i_1, i_2) + v(n_1, n_2) \tag{3.95}$$

For an $N_1 \times N_2$ observed blurred image, Eqn. (3.95) yields $N_1 \times N_2$ coupled linear equations with $N_1 \times N_2$ unknowns $s(i_1, i_2)$, assuming that the PSF is known. Hence, the restoration problem can be formulated as solving an inconsistent set of $N_1 \times N_2$ simultaneous equations in the presence of noise. In the case of LSI blurs, the system matrix is block-Toeplitz; thus, the $N_1 \times N_2$ equations can be decoupled, under the block-circulant approximation, by applying the 2D discrete Fourier transform. In the LSV case, however, the system matrix is not block-Toeplitz, and fast methods in the transform domain are not applicable.

Based on these models, we can pose the following image restoration problems:

1. Shift-invariant restoration: Given the image formation model (3.92) and the shift-invariant PSF $h(n_1, n_2)$, find an estimate $\hat{s}(n_1, n_2)$ of the image.
2. Shift-varying restoration: Given the image formation model (3.95) and the space-varying PSF $h(n_1, n_2; i_1, i_2)$, find an estimate $\hat{s}(n_1, n_2)$ of the image.

The problem is called blind-image restoration (see Section 3.6.3) if the PSF is unknown. The intraframe image-restoration problem is extended to multi-frame video restoration in Chapter 6, where it is formulated as: given a sequence of correlated frames find an estimate of all frames processing them simultaneously.

### 3.6.2 Restoration of Images Degraded by Linear Space-Invariant Blurs

Image restoration is defined as the process of undoing image blurring (smearing) based on a mathematical model of its formation. Because the model (3.92) is generally not invertible, restoration algorithms employ regularization techniques by using *a priori* information about the ideal image. Many methods exist for image restoration depending on the amount and nature of the *a priori* information used. The classical techniques are pseudo-inverse filtering, Wiener and constrained least-squares (CLS) filtering, the constrained iterative method [Sch 81], the maximum *a posteriori* probability (MAP) estimation method [Tru 79], and the projection onto convex sets (POCS) method [Tru 84, Tru 85]. In this section, we first discuss pseudo-inverse filtering and CLS/Wiener filtering. The reader is referred to [Sez 90] for a review of classical image-restoration methods. We then introduce restoration methods based on sparse-image modeling, which have recently been proposed [Ela 10].

**Pseudo-Inverse Filtering**

If we neglect the noise in the degradation model (3.92), i.e., if the vector $\mathbf{y}$ lies in the column space of the matrix $\mathbf{H}$ and the matrix $\mathbf{H}$ is invertible, an exact solution can be found by direct inversion:

$$\hat{\mathbf{s}} = \mathbf{H}^{-1}\mathbf{y} \tag{3.96}$$

where $\hat{\mathbf{s}}$ denotes an estimate of the ideal image. The operation (3.96) is known as inverse filtering. However, because inverse filtering using (3.96) requires a huge matrix inversion, it is usually implemented in the frequency domain.

Taking the 2D (spatial) Fourier transforms of both sides of (3.92), ignoring the noise term, we have

$$Y(f_1,f_2) = H(f_1,f_2)\, S(f_1,f_2)$$

Then, inverse filtering can be expressed in the frequency domain as

$$\hat{S}(f_1,f_2) = \frac{Y(f_1,f_2)}{H(f_1,f_2)} \tag{3.97}$$

The implementation of (3.97) requires sampling of the frequency variables $(f_1,f_2)$, which corresponds to computing the discrete Fourier transform (DFT) of the respective signals. Note that sampling in the frequency domain results in spatial-domain aliasing given by (3.69).

The relationship between (3.96) and (3.97) can also be shown by diagonalizing the matrix H. Because $N^2 \times N^2$ matrix $\mathbf{H}$ is block-Toeplitz for spatially shift-invariant blurs, it can be diagonalized under the block-circulant approximation as $\mathbf{H} = \mathbf{W}\Lambda\mathbf{W}^{-1}$, where $\mathbf{W}$ is the $N^2 \times N^2$ 2D-DFT matrix that contains $N^2$ partitions of size $N \times N$. The *im*th partition of $\mathbf{W}$ is in the form $W_{im} = e^{-j\frac{2\pi}{N}im} e^{-j\frac{2\pi}{N}ln}$, $l, n = 0, \ldots, N-1$, and $\Lambda$ is a diagonal matrix formed by the eigenvalues of $\mathbf{H}$, which are given by the 2D-DFT of $h[n_1, n_2]$ [Gon 07]. Then, premultiplying both sides of (3.96) by $\mathbf{W}^{-1}$, and inserting the term $\mathbf{W}\mathbf{W}^{-1}$ between $\mathbf{H}^{-1}$ and $\mathbf{y}$, we have

$$\mathbf{W}^{-1}\hat{\mathbf{s}} = (\mathbf{W}^{-1}\mathbf{H}^{-1}\,\mathbf{W})\mathbf{W}^{-1}\mathbf{y}$$

$$\hat{S}(k_1, k_2) = \frac{Y(k_1, k_2)}{H(k_1, k_2)}$$

where multiplication by $\mathbf{W}^{-1}$ computes the 2D-DFT, and $k_1$ and $k_2$ denote the discretized spatial-frequency variables; hence, the result is equivalent to (3.97).

Inverse filtering has some drawbacks. First, the matrix $\mathbf{H}$ may be singular, which means at least one of its eigenvalues $H(k_1, k_2)$ may be zero, resulting in a division by zero in the 2D-DFT implementation (3.97). Even if the matrix $\mathbf{H}$ is non-singular, i.e., $H(k_1, k_2) \neq 0$, for all $(k_1, k_2)$, the vector $\mathbf{y}$ almost never lies in the column space of the matrix $\mathbf{H}$ due to the presence of observation noise $\mathbf{v}$; thus, an exact solution for (3.96) does not exist. Then, one must resort to a least-squares (LS) solution that minimizes the norm of the residual $\mathbf{y} - \mathbf{Hs}$. The LS solution, known as pseudo-inverse filtering, given by

$$\hat{\mathbf{s}} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\,\mathbf{y} \tag{3.98}$$

exists if the columns of the matrix $\mathbf{H}$ are linearly independent. The pseudo-inverse filter can be implemented in the frequency domain by (3.97), where division by zero is defined as zero.

Deconvolution by pseudo-inversion is ill-posed due to the presence of observation noise. This is because the pseudo-inverse of the blur transfer function usually has very large magnitude near those frequencies where the blur transfer function has zeros and at high frequencies. This results in excessive noise amplification at those frequencies that can be alleviated by regularized inversion.

### Regularization by Constrained Least Squares – Wiener Deconvolution

Regularized deconvolution methods use *a priori* information about the image to roll off the transfer function of the pseudo-inverse filter near singular frequencies and at

high frequencies in an attempt to limit noise amplification. However, the regularized filter inevitably deviates from the exact inverse at those frequencies, which leads to other kinds of artifacts, known as regularization artifacts [Tek 90]. Regularization of the inversion can be achieved by deterministic optimization (constrained least squares) or by statistical estimation (Wiener filtering or MAP estimation) methods. Te reader may refer to Appendix A for an overview of regularization methods.

The CLS method aims to minimize the $l^2$-norm of the high-frequency content in the restored image $\|\mathbf{L}\ \mathbf{s}\|_2^2$, which requires it to be as smooth as possible, while ensuring the solution is consistent with the observations, i.e., satisfy the degradation model. Hence, it is formulated as a constrained optimization problem given by

$$\min_{\mathbf{s}} \|\mathbf{L}\,\mathbf{s}\|_2^2$$
$$\text{subject to } \|\mathbf{y}-\mathbf{H}\mathbf{s}\|_2^2 = \sigma_v^2 \tag{3.99a}$$

where $\mathbf{L}$ is a regularization operator and $\sigma_v^2$ denotes the variance of the noise. The operator $\mathbf{L}$ is typically chosen to have high-pass characteristics (e.g., the Laplacian filter) so the CLS method finds the smoothest image (i.e., the image with the smallest high-frequency content) which statistically complies with the observation model (3.92). This problem can be converted into an unconstrained optimization problem using the Lagrangian formulation (see Appendix A),

$$\min_{\mathbf{s}} E(\mathbf{s}), \text{ where } E(\mathbf{s}) = \|\mathbf{L}\,\mathbf{s}\|_2^2 + \lambda(\|\mathbf{y}-\mathbf{H}\mathbf{s}\|_2^2 - \sigma_v^2) \tag{3.99b}$$

and $\lambda$ is the Lagrange multiplier. Differentiating $E(\mathbf{s})$ with respect to $\mathbf{s}$ and setting the result equal to zero, we get

$$\Phi(\hat{\mathbf{s}}) = \frac{1}{2}\nabla_{\mathbf{s}} E(\mathbf{s}) = \mathbf{L}^{\mathrm{T}}\mathbf{L}\,\hat{\mathbf{s}} - \lambda \mathbf{H}^{\mathrm{T}}(\mathbf{y}-\mathbf{H}\hat{\mathbf{s}}) = \mathbf{0} \tag{3.100a}$$

Solving for $\hat{\mathbf{s}}$, we obtain the CLS filter, which can be expressed as

$$\hat{\mathbf{s}} = (\mathbf{H}^{\mathrm{T}}\mathbf{H} + \alpha \mathbf{L}^{\mathrm{T}}\mathbf{L})^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{y} \tag{3.100b}$$

where $\alpha = 1/\lambda$ is called the regularization parameter, which controls the tradeoff between smoothness of the solution and fidelity to the observations. It can be readily seen that the pseudo-inverse filter (3.98) is a special case of the CLS filter with $\alpha = 0$. Direct implementation of (3.100) requires a huge matrix inversion. There are

two approaches to an efficient implementation: DFT domain implementation and spatial-domain successive iterative approximations.

*DFT Domain Implementation*

We premultiply both sides of (3.100b) by the 2D-DFT operator $\mathbf{W}^{-1}$ to arrive at the frequency-domain filter expression after some matrix manipulations:

$$\hat{S}(k_1,k_2) = \frac{H^*(k_1,k_2)}{|H(k_1,k_2)|^2 + \alpha\,|L(k_1,k_2)|^2}\;Y(k_1,k_2) \tag{3.101}$$

where $L(k_1, k_2)$ denotes the eigenvalues of the regularization operator $\mathbf{L}$. Eqn. (3.101) can be efficiently implemented in the DFT domain. The operator $\mathbf{L}$ can be defined in terms of a 2D shift-invariant generating kernel (impulse response), and $L(k_1, k_2)$ is the 2D-DFT of this kernel.

*Successive Iterative Approximations*

Successive iterations is a gradient-descent method (covered in Appendix C) to optimize $E(\mathbf{s})$ by taking a step in the negative direction of the gradient $\Phi(\hat{\mathbf{s}})$. The solution is given by

$$\hat{\mathbf{s}}_{k+1} = \hat{\mathbf{s}}_k - \beta\;\Phi(\hat{\mathbf{s}}_k)$$

where $\beta$ is called the step size, with the initial condition $\hat{\mathbf{s}}_k$=0. Clearly a root of $\Phi(\hat{\mathbf{s}})$ is a fixed point of the iteration. Substituting (3.100a) into the successive approximation iteration yields

$$\hat{\mathbf{s}}_{k+1} = \hat{\mathbf{s}}_k + \beta\;\mathbf{H}^{\mathrm{T}}\mathbf{y} - \beta(\mathbf{H}^{\mathrm{T}}\mathbf{H} + \alpha\mathbf{L}^{\mathrm{T}}\mathbf{L})\hat{\mathbf{s}}_k = \beta\;\mathbf{H}^{\mathrm{T}}\mathbf{y} + [\mathbf{I} - \beta(\mathbf{H}^{\mathrm{T}}\mathbf{H} + \alpha\mathbf{L}^{\mathrm{T}}\mathbf{L})]\hat{\mathbf{s}}_k \tag{3.102}$$

which for $\alpha$=0 reduces to the Landweber iteration that converges to the least-squares solution. With any iterative algorithm, there are two important concerns: i) does it converge, and if so, ii) what is the limiting solution? We answer these using frequency-domain analysis. Let

$$\hat{S}_{k+1}(f_1,f_2) = \beta\;H^*(f_1,f_2)Y(f_1,f_2)$$
$$+[1-\beta(|H(f_1,f_2)|^2 + \alpha\,|L(f_1,f_2)|^2)]\hat{S}_k(f_1,f_2)$$

It is easy to show that the frequency response of the filter at the $k$th iteration is

$$T_k(f_1,f_2) = \epsilon \sum_{m=0}^{k-1} [1-\beta(|H(f_1,f_2)|^2 + \alpha |L(f_1,f_2)|^2)]^m H^*(f_1,f_2)$$

Hence, if

$$|1-\beta(|H(f_1,f_2)|^2 + \alpha |L(f_1,f_2)|^2)| < 1 \tag{3.103}$$

then

$$\lim_{k\to\infty} T_k(f_1,f_2) = \frac{H^*(f_1,f_2)}{|H(f_1,f_2)|^2 + \alpha |L(f_1,f_2)|^2}$$

Therefore, the iterations converge to the CLS solution provided that (3.103) is satisfied.

*Choices for the Regularization Operator*

Common choices include the identity operator, Laplacian operator, and Wiener operator. The former applies a fixed regularization parameter independent of image-frequency components, where we have $\mathbf{L} = \mathbf{I}$, and $\mathbf{I}$ is the identity matrix. The 2D-generating kernel in this case is the Kronecker delta function $l[n_1, n_2] = \delta[n_1, n_2]$ and $L(k_1, k_2) = 1$. The Laplacian operator applies frequency-dependent regularization, where the frequency response of the CLS filter rolls off at high frequencies. The generating kernel is a discrete approximation $l[n_1, n_2]$ to the Laplacian, which is discussed in Section 3.3.2. Then, $L(k_1, k_2)$ is the 2D-DFT of a particular discrete approximation $l[n_1, n_2]$.

The Wiener deconvolution filter, which can be derived by following the steps shown in Section 3.5.2 based on the observation model (3.92), is a special case of the CLS filter with $\mathbf{L}=\mathbf{R}_s^{-1}\,\mathbf{R}_v$ or $|L(k_1,k_2)|^2 = P_{vv}(k_1,k_2)/P_{ss}(k_1,k_2)$, where $\mathbf{R}_s$ and $\mathbf{R}_v$ denote the covariance matrices of the ideal image and the noise, and $P_{ss}(k_1,k_2)$ and $P_{vv}(k_1,k_2)$ denote their power spectra, respectively. Recall that the Wiener filter gives the linear minimum mean-square error (LMMSE) estimate of the ideal frame given the observation model. However, it requires *a priori* information about the image and noise in the form of their power spectra.

**Regularization by Sparse-Image Modeling**

Sparse-image models have recently been introduced as a detail-preserving alternative to the classical smoothness models for regularization of inverse problems including image restoration, super resolution, and in-painting [Ela 10]. The reader should refer to Appendix A for a discussion of sparse models.

We have studied the orthogonal wavelet transform (DWT) shrinkage method as a simple application of sparse-image modeling to image-denoising problem in Section 3.5.4. This approach has been extended to image restoration within an iterative expectation-maximization (EM) framework, which alternates between an E-step for deconvolution based on the fast Fourier transform (FFT) and a DWT-based M-step, which enforces sparsity of the solution, resulting in an efficient iterative process requiring $O(N \log N)$ operations per iteration [Fig 03].

A mathematical formulation that includes several independently developed image-restoration methods, including the EM framework, based on sparse image modeling is given by [Ela 10]. In this formulation, the sparse-representation coefficients of the restored image can be determined from

$$\hat{\alpha} = \arg\min_{\alpha} \{\lambda \|\alpha\|_p^p + \tfrac{1}{2} \| \mathbf{HD}\alpha - \mathbf{y} \|_2^2\} \tag{3.104}$$

where $\mathbf{D}\alpha$ is a sparse-image representation, $\mathbf{H}$ is the degradation matrix, and $\mathbf{y}$ denotes the noisy and blurred observations. This problem can be solved by using the framework of iterative shrinkage threshold (IST) algorithms, given by

$$\hat{\alpha}_{k+1} = \arg\min_{\alpha} \{\lambda \|\alpha\|_p^p + \tfrac{1}{2} \| \alpha - \beta_k \|_2^2\} \tag{3.105a}$$

which alternates between two simple steps: i) E-step: update $\boldsymbol{\beta}_k$ using

$$\beta_k = \hat{\alpha}_k - \gamma \mathbf{D}^{\mathrm{T}} \mathbf{H}^{\mathrm{T}} (\mathbf{HD}\hat{\alpha}_k - \mathbf{y}) \tag{3.105b}$$

which can be implemented by $O(N \log N)$ operations, and ii) M-step: a scalar shrinkage of $\beta_k$ given by Eqn. (3.82) The overall process is fast and effective. Several IST-like fast algorithms have also been proposed [Dia 07, Bec 09].

#### *Exploiting Non-Local Self-Similarities*

Sparse image modeling for image restoration has been extended to exploit non-local image self-similarities [Dan 12, Don 13b]. In particular, [Dan 12] extends BM3D image modeling to *iterative decouple deblurring* (IDD-BM3D), which aims to achieve generalized Nash equilibrium balance between how well the restored image fits the observations and the sparseness of the solution. Alternatively, [Don 13b] introduces *non-locally centralized sparse representation* based restoration, which can be solved by IST algorithms.

**Boundary Problem in Image Restoration**

Image sensors have a finite field of view. Since the values of the blurred pixels on the borders of the field of view (image) depend on some pixels outside the field of view, part of the information necessary to restore border pixels is not available. Although it seems that the effect of this problem would be limited to only border pixels, in general the impulse response of restoration filters is quite large and the effect propagates to almost all pixels in the image. We alleviate this problem as follows: In the space-domain implementations, we extrapolate the value of unavailable pixels from border pixels, usually by replication of the first/last row/column. In DFT domain implementations, because the images are assumed to be periodic, we interpolate between the left and right and top and bottom boundary pixels to estimate the unavailable pixel values.

### 3.6.3 Blind Restoration – Blur Identification

Blind restoration refers to the case where the PSF of the blur is unknown. Since both the PSF and the original image are unknown, blind restoration is an underdetermined (ill-posed) problem, which cannot be solved without strong assumptions on the PSF or the original image or both.

**Blur Identification from Zero-Crossings in the Fourier Domain**

A common model for natural images assumes that their Fourier spectrum does not have zero-crossings. It is then possible to estimate the PSF of out-of-focus and linear motion blurs from the spectrum $P_y(\omega_1, \omega_2)$ of the blurred image, because they introduce regular zero crossings, a signature of the PSF, in the spectrum of the degraded image. For linear shift-invariant blur PSFs, we can write the power spectrum of the blurred image as

$$P_y(\omega_1,\omega_2) = |H(\omega_1,\omega_2)|^2 \, P_s(\omega_1,\omega_2) \tag{3.106}$$

Assuming $P_s(\omega_1, \omega_2)$ does not have zero-crossings, the zero-crossings of $P_y(\omega_1, \omega_2)$ are due to those of $H(\omega_1, \omega_2)$. We estimate the power spectrum of the observed image in the DFT domain using the periodogram method given by

$$P_y[k_1,k_2] = \frac{1}{N_1 N_2} |Y[k_1,k_2]|^2 \tag{3.107}$$

where $Y[k_1, k_2]$ is the DFT of the noise-contaminated $N_1 \times N_2$ image. The cepstrum is defined as the inverse discrete Fourier transform of the logarithm of the image power spectrum, given by

$$c[n_1, n_2] = -\text{IDFT}(\log\{P_y[k_1, k_2]\}) \tag{3.108}$$

Note that it is easier to identify zero crossings in the cepstrum domain, since the logarithm enhances the visibility of the location of the zero crossings.

**Example: Identification of Uniform Motion and Out-of-Focus Blurs**

Modeling linear motion blur by a uniform rectangular PSF, its frequency response is given by a sinc function, which has periodic zeros. In the case of an $8 \times 1$ horizontal motion blur and $128 \times 128$ DFT, there are seven zero crossings at each row of the 2D-DFT occuring at the samples $k_1 = 16$, 32, 48, 64, 80, 96, and 112. Similarly, modeling an out-of-focus blur by a uniform circular PSF, its frequency response is given by a Bessel function, which has regular zeros as a function of the radius of the circle. Then, we can estimate the radius of the PSF from the zero crossings of the power spectrum of the degraded image.

### Blur Identification Using Parametric Models

A maximum-likelihood formulation for parametric blur identification has been proposed, where the image is modeled as a 2D auto-regressive model and the blur is modeled by a FIR filter [Pav 92]. This method works especially well for blurs that do not have zeros in the frequency domain, such as Gaussian blurs.

### Blur Identification in the Image-gradient Domain

A recent approach proposes blur identification based on image gradients rather than image intensities [Fer 06]. Since both differentiation and convolution are linear operators, the effect of blur in the gradient domain can be modeled as

$$\nabla y(n_1, n_2) = h(n_1, n_2) ** \nabla s(n_1, n_2)$$

where $\nabla$ denotes the spatial-gradient operator. The *a priori* distribution of gradient of sharp natural images is modeled by a zero-mean, heavy-tailed mixture of Gaussians model, resulting in a maximum *a posteriori* probability (MAP) estimation problem to find $h(n_1,n_2)$ [Fer 06].

## 3.6.4 Restoration of Images Degraded by Space-Varying Blurs

In comparison with the amount of work on linear space-invariant (LSI) image restoration, the literature reports only a few methods for the restoration of images degraded by linear shift-varying (LSV) blurs. Nevertheless, in real-world applications degradations are often space-varying; i.e., the degrading system PSF changes with spatial location over the image. Examples of LSV blurs include motion blur when the relative motion is not parallel to the imaging plane or contains acceleration and out-of-focus blur when the scene has significant depth variation. Space-varying restoration methods include coordinate transformations [Saw 74], sectional processing [Tru 78, Tru 92], and iterative methods [Sch 81]. The coordinate transformation approach is applicable only to a special class of LSV degradations that can be transformed into an LSI degradation. In sectional methods the image is sectioned into rectangular regions, where each section is restored using a space-invariant method, such as the maximum *a posteriori* filter [Tru 78] or the modified Landweber iterative filter [Tru 92].

Here, we present the projections onto convex sets (POCS) formulation that is applicable to any shift-varying blur type and size. The POCS framework has also been used for LSI image restoration. In [Tru 84], Trussell and Civanlar use variance of the residual constraint for restoration of space-invariant blurs. However, this constraint cannot easily be extended to space-variant restoration since it involves inversion of huge matrices that would not be Toeplitz for space-varying blurs. Later, Sezan and Trussell [Sez 91] develop a general framework of prototype-image-based constraints. Although the framework is general, specific constraints proposed in [Sez 91] have been designed for space-invariant blurs.

### Overview of the POCS Method

In the POCS method, the unknown signal, **s**, is assumed to be an element of an appropriate Hilbert space. Each *a priori* information or constraint restricts the solution to a closed convex set. Thus, for $m$ pieces of information, there are $m$ closed convex sets $C_i \in H$, $i = 1,2,\ldots,m$, and $\mathbf{s} \in C_0 \equiv \cap_{i=0}^{m} C_i$ provided the intersection $C_0$ is non-empty.

Given the sets $C_i$ and their respective projection operators $\mathbf{P}_i$, the sequence generated by

$$\mathbf{s}_{k+1} = \mathbf{P}_m \mathbf{P}_{m-1} \cdots \mathbf{P}_1 \mathbf{s}_k, \quad k = 0,1,\ldots \tag{3.109a}$$

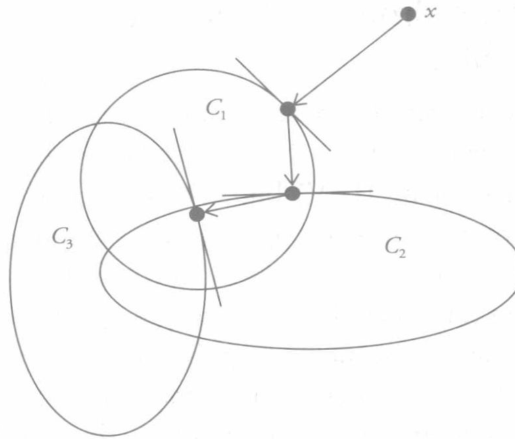



**Figure 3.27** Method of projection onto convex sets.

or, more generally, by

$$\mathbf{s}_{k+1} = \mathrm{T}_m \mathrm{T}_{m-1} \cdots \mathrm{T}_1 \mathbf{s}_k, \ k = 0,1,\ldots \tag{3.109b}$$

where $\mathbf{T}_i \equiv (1-\lambda_i)\mathbf{I} + \lambda_i \mathbf{P}_i$, $0 < \lambda_i < 2$ is the relaxed projection operator, converges weakly to a feasible solution in the intersection $C_0$ of the constraint sets. Convex sets and successive projections are illustrated in Figure 3.27. Indeed, any solution in the intersection set is consistent with the *a priori* constraints and therefore is a feasible solution. Note that $\mathbf{T}_i$ reduces to $\mathbf{P}_i$ for unity relaxation parameter, i.e., $\lambda_i = 1$. The initialization $\mathbf{s}_0$ can be arbitrarily chosen. It should be emphasized that the POCS algorithm is in general nonlinear, because the projection operations are in general nonlinear. For more detailed discussions of the fundamental concepts of the POCS, the reader is referred to [Tru 84].

### Restoration Using POCS

The POCS method can readily be applied to the space-varying restoration problem using a number of space-domain constraints that are defined on the basis of the observed image and *a priori* information about the space-varying degradation process, the noise statistics, and the ideal image itself [Ozk 94].

Assuming that the space-varying blur PSF and the statistics of noise are known, we define the following closed, convex feasible solution (constraint) sets (one for each observed image pixel):

$$C_{n_1,n_2} = \{z(i_1,i_2) \| r^z(n_1,n_2)| \leq \delta_0\} \ 0 \leq n_1 \leq N_1 - 1, \ 0 \leq n_2 \leq N_2 - 1 \tag{3.110a}$$

where

$$r^z(n_1,n_2) = y[n_1,n_2] - \sum_{i_1=0}^{N_1-1} \sum_{i_2=0}^{N_2-1} z[i_1,i_2]\, h(n_1,n_2;i_1,i_2) \tag{3.110b}$$

is the residual associated with solution $\mathbf{z}$, which is an arbitrary member of the set. The quantity $\delta_0$ is an *a priori* bound reflecting the statistical confidence with which the actual image is a member of the set $C_{n_1,n_2}$. If $\mathbf{s}$ denotes the ideal image, then $r^s(n_1,n_2) = v[n_1,n_2]$ and statistics of $r^s(n_1,n_2)$ should be identical to that of $v[n_1,n_2]$. Hence, the bound $\delta_0$ is determined from the statistics of the noise so that the ideal (actual) solution is a member of the set within a certain statistical confidence. For example, if the noise has Gaussian distribution with the standard deviation $\sigma_v$, $\delta_0$ is set equal to $c\,\sigma_v$, where $c \geq 0$ is determined by an appropriate statistical confidence bound (e.g., $c$=3 for 99% confidence). The bounded residual constraint enforces the estimate to be consistent with the observed image. In other words, at each iteration, the estimate is constrained such that at every pixel $(n_1,n_2)$, the absolute value of the residual between the observed image value $y[n_1,n_2]$ and the pixel value obtained at $(n_1,n_2)$ by simulating the imaging process using that estimate is required to be less than a predetermined bound.

The projection $P_{n_1,n_1}\{x[i_1,i_2]\}$ of an arbitrary $x[i_1,i_2]$ onto $C_{n_1,n_2}$ can be defined as [Tru 84]:

$$P_{n_1,n_2}\{x[i_1,i_2]\} =$$
$$\begin{cases} x[i_1,i_2] + \dfrac{r^x(n_1,n_2)-\delta_0}{\sum_{o_1}\sum_{o_2} h^2(n_1,n_2;o_1,o_2)} h(n_1,n_2;i_1,i_2) & \text{if } r^x(n_1,n_2) > \delta_0 \\ x[i_1,i_2] & -\delta_0 \leq r^x(n_1,n_2) \leq \delta_0 \\ x[i_1,i_2] + \dfrac{r^x(n_1,n_2)+\delta_0}{\sum_{o_1}\sum_{o_2} h^2(n_1,n_2;o_1,o_2)} h(n_1,n_2;i_1,i_2) & \text{if } r^x(n_1,n_2) < -\delta_0 \end{cases} \tag{3.111}$$

Additional constraints, such as bounded energy, amplitude, and limited support, can be utilized to improve the results. For example, the amplitude constraint can be defined as

$$C_A = \{z(i_1,i_2) \mid \alpha \leq z(i_1,i_2) \leq \beta \text{ for } 0 \leq i_1 \leq N_1 - 1,\ 0 \leq i_2 \leq N_2 - 1\} \tag{3.112}$$

with amplitude bounds of $\alpha$=0 and $\beta$=255. The projection $\mathbf{P}_A$ onto the amplitude constraint $C_A$ is defined as

$$\mathbf{P}_A\{x[i_1,i_2]\} = \begin{cases} 0 & \text{if } x[i_1,i_2] < 0 \\ x[i_1,i_2] & \text{if } 0 \le x[i_1,i_2] \le 255 \\ 255 & \text{if } x[i_1,i_2] > 255 \end{cases} \tag{3.113}$$

Given the above projections, an estimate $\hat{s}_k(i_1, i_2)$ of the ideal image $s(i_1, i_2)$ is obtained iteratively as

$$\hat{s}_{k+1}[i_1,i_2] = \mathbf{T}_A \mathbf{T}_{N_1-1,N_2-1} \mathbf{T}_{N_1-2,N_2-1} \cdots \mathbf{T}_{0,0}\{\hat{s}_k[i_1,i_2]\} \tag{3.114}$$

where $k = 0,1,\ldots,$ $0 \le i_1 \le N_1 - 1$, $0 \le i_2 \le N_2 - 1$, and $\mathbf{T}$ denotes the generalized projection operator defined in (3.111), and the observed noisy and blurred image $y[i_1, i_2]$ can be taken as the initial estimate $\hat{s}_0(i_1, i_2)$. Note that in every iteration cycle $k$, each of the projection operators $\mathbf{T}_{n_1,n_2}$ is used once, but the order in which they are implemented is arbitrary.

### 3.6.5 Image In-Painting

Image in-painting refers to reconstruction/concealment of damaged or missing areas in an image in a way that cannot be distinguished from the original. It has many applications including removal of scratches, text overlays or an undesired object, concealment of image transmission losses such as packet losses, and disocclusion in image-based rendering of intermediate virtual camera viewpoints. The term originates from manual restoration of artwork, such as paintings or old photographs, by skilled artists and is also known as "retouching."

Image in-painting problem can be modeled as

$$\mathbf{y} = \mathbf{M}\mathbf{s} + \mathbf{v}$$

where $\mathbf{M}$ is an $N \times N$ diagonal mask matrix with ones for existing pixels and zeros for the missing ones along the diagonal. Digital in-painting differs from image interpolation, which often refers to re-sampling of a uniformly sampled data set. In-painting is clearly an ill-posed problem with many plausible solutions. The objective is that the reconstructed regions look natural and physically realistic to the human eye. In-painting methods can be classified as diffusion-based, statistics and examplar-based, and sparse-representation-based methods [Gui 14].

Diffusion-based in-painting introduces smoothness priors by using parametric models or partial differential equations (PDEs) to propagate (or diffuse) local

structures, such as straight lines or curves, from the borders of a missing patch toward the interior. Many variants exist using different linear, nonlinear, isotropic, or anisotropic models to favor propagation in particular directions or to take into account the curvature of structures present in a local neighborhood. These methods are well suited for in-painting small regions. They avoid generating unconnected edges; however, they tend to blur textures and large areas.

Statistics and examplar-based methods exploit image self-similarity and statistical models. The statistics of image patches or textures are assumed to be stationary (for random textures) or homogeneous (for regular patterns). Local region-growing methods grow texture one pixel or one patch at a time, while maintaining coherence with nearby pixels. The principle of examplar-based methods is searching for a patch that is most similar to the known part of the missing patch, and copy and paste the central pixel for pixel-based approaches or a set of pixels for patch-based approaches.

With the recent growing interest in low-rank and sparse-image representations, sparse priors have also been used for solving the in-painting problem. The image is assumed to be composed of low-rank and sparse components in a given basis, e.g., DCT or wavelets. Known and missing parts of the image are assumed to share the same representation. The missing region is synthesized as a sparse linear combination of elements from an overcomplete dictionary [Gui 14].

Example-based and sparse-based methods are better suited to fill large missing texture areas than diffusion-based methods. Hybrid methods, which combine structural (geometrical) and textural components, have also emerged.

## References

[Arc 91] Arce, G. R., "Multistage order statistic filters for image sequence processing," IEEE Trans. Signal Proc., vol. 39, pp. 1146–1163, 1991.

[Ata 80] Ataman, E., V. K. Aatre, and K. M. Wong, "A fast method for real-time median filtering," IEEE Trans. Acoust. Speech Sign. Proc., vol. 28, pp. 415–421, Aug. 1980.

[Bak 02] Baker, S. and T. Kanade, "Limits on super-resolution and how to break them," IEEE Trans. Patt. Anal. Mach. Intell., vol. 24, no. 9, pp. 1167–1183, Sept. 2002.

[Bar 04] Barash, D. and D. Comaniciu, "A common framework for nonlinear diffusion, adaptive smoothing, bi-lateral filtering and mean shift," Image and Video Computing, vol. 22, pp. 73–81, 2004.

[Bec 09] Beck, A. and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," SIAM J. Imag. Sci., vol. 2, no. 1, pp. 183–202, 2009.

[Bov 00] Bovik, Al, ed., *Handbook of Image and Video Processing,* San Diego, CA: Academic, 2000.

[Bua 05] Buades, A., B. Coll, and J. M. Morel, "A non-local algorithm for image denoising," IEEE Conf. On Computer Vision and Pattern Recognition (CVPR), vol. 2, pp. 60–65, 2005.

[Can 86] Canny, J., "A computational approach to edge detection," IEEE Trans. on Patt. Anal. Mach. Intel., vol. 8, no. 6, pp. 679–698, 1986.

[Cha 85] Chan, P. and J. S. Lim, "One-dimensional processing for adaptive image restoration," IEEE Trans. Acoust. Speech and Sign. Proc., vol. 33, pp. 117–126, Feb. 1985.

[Cha 00] Chang, S. G., B. Yu and M. Vetterli, "Adaptive wavelet thresholding for image denoising and compression," IEEE Trans. on Image Processing, vol. 9, no. 9, pp. 1532–1546, Sept 2000.

[Cha 10] Chatterjee, P. and P. Milanfar, "Is denoising dead?" IEEE Trans. on Image Processing, vol. 19, no. 4, pp. 895–911, April 2010.

[Cro 83] Crochiere, Ronald E. and Lawrence R. Rabiner, *Multirate Digital Signal Processing,* Englewood Cliffs, NJ: Prentice Hall, 1983.

[Dab 07] Dabov, K., A. Foi, V. Katkovnik, and K.O. Egiazarian, "Image de-noising by sparse 3-D transform-domain collaborative filtering," IEEE Trans. Image Process., vol. 16, no. 8, pp. 2080–2095, Aug. 2007.

[Dan 12] Danielyan, A., V. Katkovnik, and K. Egiazarian, "BM3D frames and variational image deblurring," IEEE Trans. on Image Processing, vol. 21, no. 4, pp. 1715–1728, April 2012.

[Dau 92] I. Daubechies, "Ten lectures on wavelets," CBMS-NSF Regional Conference Series in Applied Mathematics, vol. 61, 1992.

[Dav 75] Davis, L. S., "A survey of edge-detection techniques," Comp. Graph. and Image Proc., vol. 4, pp. 248–270, 1975.

[Dia 07] Bioucas-Dias, J. and M. Figueiredo, "A new TwIST: Two-step iterative shrinkage/ thresholding algorithms for image restoration," IEEE Trans. on Image Process., vol. 16, pp. 2992–3004, 2007.

[Don 95a] Donoho, D. L., "Denoising via soft thresholding," IEEE Trans. on Information Theory, vol. 41, pp. 613–627, May 1995.

[Don 95b] Donoho, D. L., and I. M. Johnstone, "Adapting to unknown smoothness via wavelet shrinkage," Jour. of the American Statistical Association, vol. 90 (432), pp. 1200–1224, 1995.

[Don 13a] Dong, W., L. Zhang, R. Lukac, and G. Shi, "Sparse representation based image interpolation with nonlocal autoregressive modeling," IEEE Trans. on Image Proc., vol. 22, no. 4, pp. 1382–1394, Apr. 2013.

[Don 13b] Dong, W., L. Zhang, G. Shi, X. Li, "Nonlocally centralized sparse representation for image restoration," IEEE Trans. on Image Proc., vol. 22, no. 4, pp. 1620–1630, Apr. 2013.

[Dur 02] Durand, F., and J. Dorsey, "Fast bi-lateral filtering for the display of high dynamic range images," ACM Trans. on Graphics (SIGGRAPH 2002), vol. 21 no. 3, pp. 257–266, July 2002.

[Ela 10] M. Elad, M. A. T. Figueiredo, and Y. Ma, "On the role of sparse and redundant representations in image processing," Proceedings of the IEEE, vol. 98, no. 6, pp. 972–982, June 2010.

[Fer 06] Fergus, R., B. Singh, A. Hertzmann, S. T. Roweis, and W. T. Freeman, "Removing camera shake from a single photograph," ACM SIGGRAPH, pp. 784–794, 2006.

[Fig 03] Figueiredo, M., and R. Nowak, "An EM algorithm for wavelet-based image restoration," IEEE Trans. on Image Processing, vol. 12, no. 8, pp. 906–916, 2003.

[Fod 03] Fodor, I. K., and C. Kamath, "Denoising through wavelet shrinkage: An empirical study," Jour. of Electronic Imaging, vol. 12, no. 1, Jan. 2003.

[Fre 02] Freeman, W. T., T. R. Jones, and E. C. Pasztor, "Example-based super-resolution," IEEE Computer Graphics and Applications, vol. 22, pp. 56–65, March/April 2002.

[Fun 04] Funt, B., F. Ciurea, and J. J. McCann, "Retinex in MATLAB," *J. of Electronic Imaging,* vol. 13, no. 1, pp. 48–57, Jan 2004.

[Gem 84] Geman, S., and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," IEEE Trans. on Patt. Anal. Mach. Intell. 6 (6), pp. 721–74, 1984.

[Gon 07] Gonzalez, Rafael C., and Richard E. Woods, *Digital Image Processing, Third Edition*, Upper Saddle River, NJ: Prentice Hall, 2007.

[Gui 14] Guillemot, C., and O. L. Meur, "Image in-painting: Overview and recent advances," IEEE Signal Processing Magazine, vol. 31, no. 1, pp. 127–144, Jan. 2014.

[Gun 05] Gunturk, B., J. Glodtzbach, Y. Altunbasak, R.S. Schafer, R. Mersereau, "De-Mosaicking: Color filter array interpolation," IEEE Signal Process. Mag., vol. 22, no. 1, pp. 44–54, Jan. 2005.

[Han 13] Han, J.-W., Suryanto, J.-H. Kim, S. Sull, and S.J. Ko, "New edge-adaptive image interpolation using anisotropic Gaussian filters," Dig. Signal Proc., vol. 23, pp. 110–117, 2013.

[Har 88] Harris, C., and M. Stephens, "A combined corner and edge detector," Proc. of the 4th Alvey Vision Conference, pp. 147–151, 1988.

[Hue 86] Huertas, A., and G. Medione, "Detection of intensity changes with subpixel accuracy using Laplacian of Gaussian masks," IEEE Trans. on Patt. Anal. Mach. Intel., vol. 8, no. 5, pp. 651–664, 1986.

[Kat 10] Katkovnik, V., A. Foi, K. Egiazarian, and J. Astola, "From local kernel to nonlocal multiple-model image denoising," Int. J. Computer Vision, vol. 86, no. 1, pp. 1–32, Jan. 2010.

[Key 81] Keys, R. G., "Cubic convolution interpolation for digital image processing," IEEE Trans. Acoust. Speech Sign. Proc., vol. ASSP-29, pp. 1153–1160, Dec. 1981.

[Kim 10] Kim, K. I., and Y. Kwon, "Single-image super-resolution using sparse regression and natural image prior," IEEE Trans. on Patt. Anal. Mach. Intel., vol. 32, no. 6, pp. 1127–1133, June 2010.

[Kua 85] Kuan, D. T., A. A. Sawchuk, T. C. Strand, and P. Chavel, "Adaptive noise smoothing filter for images with signal-dependent noise," IEEE Trans. on Patt. Anal. Mach. Intel., vol. 7, pp. 165–177, Mar. 1985.

[Lee 80] Lee, J. S., "Digital image enhancement and noise filtering by use of local statistics," IEEE Trans. on Patt. Anal. Mach. Intel., vol. 2, pp. 165–168, Mar. 1980.

[Li 01] Li, X. and M. Orchard, "New edge-directed interpolation," IEEE Trans. on Image Processing, vol. 10, no. 10, pp. 1521–1527, Oct. 2001.

[Low 04] Lowe, D. G., "Distinctive image features from scale-invariant keypoints," Int. Jour. of Computer Vision, vol. 60, no. 2, pp. 91–110, 2004.

[Lui 07] Luisier, F., T. Blu, and M. Unser, "A new SURE approach to image denosing: interscale orthonormal wavelet thresholding," IEEE Trans. on Image Proc., vol. 16, no. 3, pp. 593–606, 2007.

[McC 04] McCann, J. J., "Capturing a black cat in shade: past and present of Retinex color appearance models," *J. Electronic Imaging,* vol. 13, no. 1, pp. 36–47, Jan 2004.

[McD 81] McDonnell, M.J., "Box filtering techniques," Computer Graphics and Image Processing, vol. 17, p. 65, 1981.

[Men 11] Menon, D., and G. Calvagno, "Color image demosaicking: An overview," Signal Processing: Image Communication, vol. 26, no. 8–9, pp. 518–533, Oct. 2011.

[Mil 13] Milanfar, P., "A tour of modern image filtering: New insights and methods, both practical and theoretical," IEEE Signal Proc. Mag., vol. 30, no. 1, pp. 106–128, 2013.

[Mit 06] Mitra, S. K., *Digital Signal Processing, Third Edition*, New York, NY: McGraw Hill, 2006.

[Opp 68] Oppenheim, A.V., R.W. Schafer and T.G. Stockham "Nonlinear filtering of multiplied and convolved signals," Proc. of the IEEE, vol. 56, no. 8, pp. 1264–1291, Aug. 1968.

[Ozk 94] Ozkan, M. K., A. M. Tekalp, and M. I. Sezan, "POCS-based restoration of space-varying blurred images," IEEE Trans. on Image Proc., vol. 3, pp. 450–454, July 1994.

[Par 08] Paris, S., P. Kornprobst, J. Tumblin, and F. Durand, "Bi-lateral filtering: Theory and applications," Foundations and Trends in Comp. Graphics and Vision, vol. 4, no. 1, pp. 1–73, 2008.

[Pav 92] Pavlovic, G., and A.M. Tekalp, "Maximum likelihood parametric blur identification based on a continuous spatial domain model," IEEE Trans. Image Proc., vol. 1, pp. 496–504, Oct. 1992.

[Pel 82] Peli, T., and J. S. Lim, "Adaptive filtering for image enhancement," Optical Engineering, vol. 21, pp. 108–112, 1982.

[Rah 04] Rahman, Z., D. J. Jobson, and G. A. Woodell, "Retinex processing for automatic image enhancement," *Journal of Electronic Imaging*, vol. 13, no. 1, Jan. 2004.

[Rei 02] Reinhard, E., M. Stark, P. Shirley, and J. Ferwerda, "Photographic tone reproduction for digital images," ACM Trans. on Graphics (SIGGRAPH 2002), vol. 21, no. 3, pp. 267–276, July 2002.

[Saw 74] Sawchuk, A., "Space-variant image restoration by coordinate transformations," J. Opt. Soc. Am., vol. 60, pp. 138–144, Feb. 1974.

[Sch 81] Schafer, R. W., R. M. Mersereau, and M. A. Richards, "Constrained iterative restoration algorithms," Proc. of the IEEE, vol. 69, pp. 314–332, Apr. 1981.

[Sch 94] Schultz, R. R., and R. L. Stevenson, "A Bayesian approach to image expansion for improved definition," IEEE Trans. Image Proc., vol. 3, pp. 233–242, May 1994.

[Sez 90] Sezan, M. I., and A. M. Tekalp, "Survey of recent developments in digital image restoration," Optical Engineering, vol. 29, pp. 393–404, 1990.

[Sez 91] Sezan, M. I. and H. J. Trussell, "Prototype image constraints for set-theoretic image restoration," IEEE Trans. Signal Proc., vol. 39, pp. 2275–2285, Oct. 1991.

[Sha 01] Shapiro, L. G., and G. C. Stockman, *Computer Vision.* Upper Saddle River, NJ: Prentice Hall, 2001.

[Shi 94] Shi, J., and C. Tomasi, "Good features to track," IEEE Conf. Comp. Vision and Patt. Recog. (CVPR), Seattle, WA, June 1994.

[Tak 07] Takeda, H., S. Farsiu, and P. Milanfar, "Kernel regression for image processing and reconstruction," IEEE Trans. on Image Processing, vol. 16, no. 2, pp. 349–366, Feb. 2007.

[Tek 90] Tekalp, A. M., and M. I. Sezan, "Quantitative analysis of artifacts in linear space invariant image restoration," Multidim. Systems and Signal Processing, vol. 1, no. 2, pp. 143–177, 1990.

[Tom 98] Tomasi, C., and R. Manduchi, "Bi-lateral filtering for gray and color images," Proc. of the IEEE Int. Conf. on Computer Vision, Bombay, India, 1998.

[Tru 78] Trussell, H. J. and B. R. Hunt, "Image restoration of space-variant blurs by sectional methods," IEEE Trans. Acoust., Speech and Signal Proc., vol. 26, pp. 608–609, 1978.

[Tru 79] Trussell, H. J., and B. R. Hunt, "Improved methods of maximum a posteriori restoration," IEEE Trans. on Comput., vol. 27, no 1, pp. 57–62, 1979.

[Tru 84] Trussell, H. J., and M. Civanlar, "Feasible solutions in signal restoration," IEEE Trans. on Acoust. Speech Sign. Proc., vol. 32, pp. 201–212, 1984.

[Tru 85] Trussell, H. J., and M. Civanlar, "The Landweber iteration and projection onto convex sets," IEEE Trans. Acoust. Speech Sign. Proc., vol. 33, pp. 1632–1634, Dec. 1985.

[Tru 92] Trussell, H. J., and S. Fogel, "Identification and restoration of spatially variant motion blurs in sequential images," IEEE Trans. on Image Proc., vol. 1, pp. 123–126, Jan. 1992.

[Wan 07] Wang, Q., and R. K. Ward, "A new orientation-adaptive interpolation method," IEEE Trans. on Image Processing, vol. 16, no. 4, pp. 889–900, April 2007.

[Yin 96] Yin, L., R. Yang, M. Gabbouj, and Y. Neuvo, "Weighted median filters: a tutorial," IEEE Trans. Circ. and Syst. II: Analog & Digital Sign. Proc., vol. 43, no. 3, pp. 155–192, 1996.

# Exercises

## Problem Set 3

**3.1** At how many standard deviations does a Gaussian fall to 5% of its peak value? On the basis of this, suggest a suitable kernel size parameter $N$ for a Gaussian filter

$$h(n_1, n_2) = Ke^{-\left(\frac{n_1^2+n_2^2}{2\sigma^2}\right)} \quad -N \le n_1 \le N, -N \le n_2 \le N$$

given the value of $\sigma$. State how to determine the parameter $K$.

**3.2** Find the frequency response of the zero-order hold filter (3.16) and the linear interpolation filter (3.18) for $L = 2$. Compare them with that of the ideal (band-limited) interpolation filter.

**3.3** In this problem, we design 1D and 2D cubic-spline interpolation filters. We first design a continuous-time cubic spline reconstruction filter that approximates the ideal low-pass filter with the cutoff frequency $f_c = 1/2$. The impulse response $h(t)$ of a filter that uses four neighboring samples at any time $t$ can be expressed as

$$h(t) = \begin{cases} a_3|t|^3 + a_2|t|^2 + a_1|t| + a_0 & 0 \leq |t| < 1 \\ b_3|t|^3 + b_2|t|^2 + b_1|t| + b_0 & 1 \leq |t| < 2 \\ 0 & 2 \leq |t| \end{cases}$$

The design criteria are: $h(t) = 0$ for all integer $t$ (i.e., original sample points), except $t = 0$ where $h(0) = 1$, and the slope of the impulse response $dh(t)/dt$ must be continuous across original sample points.

**a.** Show that the design criteria are met if the eight unknown coefficients $a_i$ and $b_i$, $i = 0, \ldots, 3$ satisfy the following seven equations:

$$\begin{aligned} a_0 &= 1 \\ a_1 &= 0 \\ a_3 + a_2 &= -1 \\ b_3 + b_2 + b_1 + b_0 &= 0 \\ 8b_3 + 4b_2 + 2b_1 + b_0 &= 0 \\ 12b_3 + 4b_2 + b_1 &= 0 \\ 3a_3 + 2a_2 &= 3b_3 + 2b_2 + b_1 \end{aligned}$$

**b.** Let $b_3$ be a free parameter, and express all other parameters in terms of $b_3$. Determine a range of values for $b_3$ such that

$$\left.\frac{d^2h(t)}{dt^2}\right|_{t=0} < 0 \text{ and } \left.\frac{d^2h(t)}{dt^2}\right|_{t=1} > 0$$

**c.** A digital cubic spline filter can be obtained by sampling $h(t)$. Show that for interpolation by a factor of $L$, the impulse response of the cubic-spline filter has $4L + 1$ taps. Determine the impulse response for $L = 2$ and $b_3 = -1$.

**d.** Design a 2D separable cubic spline interpolation filter with the same parameters.

**3.4** Consider the following image, where each pixel value is represented by 6 bits, i.e., pixel values range between 0 and 63.

| | | | | |
|---|---|---|---|---|
| 0 | 20 | 15 | 15 | 20 |
| 10 | 20 | 30 | 0 | 15 |
| 30 | 30 | 25 | 25 | 0 |
| 20 | 35 | 10 | 10 | 0 |
| 15 | 0 | 35 | 25 | 20 |

**a.** Find $\alpha$ and $\beta$ in order to stretch the dynamic range of this image to the range [5,60] using the automatic gain control (AGC) method. Also, find the output image.

**b.** Apply histogram equalization to the above image and compare the result with part (a).

**3.5** The horizontal Sobel filter is given by

| | | |
|---|---|---|
| −1 | 0 | 1 |
| −2 | 0 | 2 |
| −1 | 0 | 1 |

Suppose it is applied following a smoothing filter:

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 1 | 1 | 1 |

Find the impulse response of the combined smoothing and Sobel filter.

**3.6** In the Wiener filter given by Eqn. (3.68), how would you estimate the power spectrum of the original image $P_{ss}(f_1,f_2)$ and the noise $P_{vv}(f_1,f_2)$?

**3.7** Let $\mathbf{s}$ denote the lexicographic ordering of pixel intensities in an image. Show that convolution by an FIR filter kernel $h[n_1,n_2]$ can be expressed as $\mathbf{y}=\mathbf{Hs}$, where $\mathbf{H}$ is block-Toeplitz and block-circulant for linear and circular convolution, respectively. Write the elements of $\mathbf{H}$ in terms of $h[n_1,n_2]$.

**3.8** Show that a block-circulant matrix can be diagonalized by a 2D-DFT.

**3.9** Suppose we have a $7 \times 1$ (pixels) uniform motion blur; i.e., $h[n_1, n_2]$ is a 1-D boxcar function that is seven pixels long. How many elements of $H(u_1, u_2)$ in Eqn. (3.91) are exactly zero, assuming a $256 \times 256$ DFT? Repeat for an $8 \times 1$ (pixels) uniform motion blur. Show, in general, that exact zeros will be encountered when the DFT size is an integer multiple of the blur size.

**3.10** Show that the CLS filter (3.101) becomes a Wiener filter when $\mathbf{L} = \mathbf{R}_s^{-1}\mathbf{R}_v$.

**3.11** The term $\alpha|L(u_1, u_2)|^2$ in Eqn. (3.101) limits noise amplification; however, it causes a different type of artifact known as a "regularization artifact" due to deviation of the regularized filter from the exact inverse. Provide a quantitative analysis of the tradeoff between noise amplification and regularization artifacts. (Hint: see [Tek 90].)

**3.12** Iterative signal and image restoration based on variations of the Landweber iteration has been heavily examined [Sch 81]. Discuss the relationship between the Landweber iterations and the POCS method. (Hint: see [Tru 85].)

## MATLAB Exercises

**3.1** Bi-lateral Filter: Implement the bi-lateral filter defined by Eqns. (3.6) and (3.7):

a. Comment on how filter kernel and output varies for different values of $N$, $\sigma_p^2$, and $\sigma_q^2$.

b. Compare filter kernel and output with a Gaussian filter with the same parameters.

c. Discuss how to implement fast bi-lateral filtering.

**3.2** Gaussian and Laplacian Pyramids (Decimation and Interpolation):

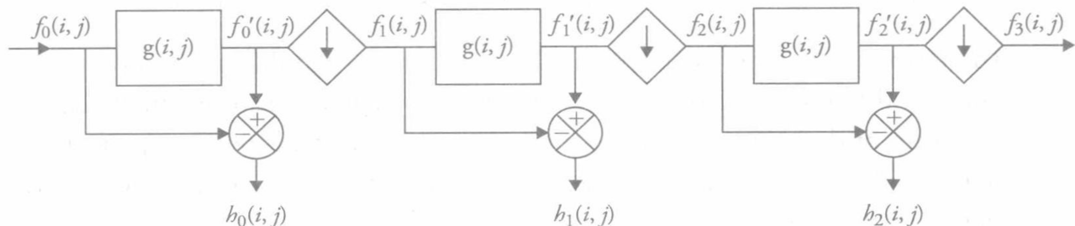
a. To construct a Gaussian pyramid:

i. Define a $7 \times 7$ circularly symmetric Gaussian filter to construct a *3-level* Gaussian pyramid. First, apply the Gaussian filter to the image, then subsample the output by a factor of 2 in both the horizontal and vertical directions to form the second level of the pyramid. Repeat the process to form the third level.

ii. Repeat (i) using $7 \times 7$ box-car filter, i.e., (1/49) ones (7,7). Compare pyramids formed by using the Gaussian and box filters and discuss the results.

iii. Now construct a resolution pyramid without any low-pass filtering. What is the problem with not using a filter? Explain in detail by referring to the frequency domain.

**b.** A Laplacian pyramid is constructed by taking the difference between the Gaussian filtered image and the original image to compute a difference image at each level of a Gaussian pyramid as depicted below. Thus, a *4-level* Laplacian pyramid consists of one low resolution picture and three difference images at the resolution of respective levels, denoted by $\{h_0, h_1, h_2, f_3\}$.




**i.** Perform interpolation using nearest-neighbor, bilinear, and bicubic spline interpolator filters. (Hint: Use the interp2 function in MATLAB.) Compare results.

**ii.** In theory, we can recover the full-resolution image by adding the difference images to the interpolated images at each level successively. But for the sake of this exercise, do not add the difference images. Just interpolate the low-resolution image by a factor of *4* in each direction, and compare the resulting image with the original high-resolution image. What differences do you see? Explain by referring to the frequency domain.

**3.3** Edge Detection: Take two color images. Convert these images from the RGB into the YCrCb domain:

**a.** Apply Sobel, Prewitt, and LoG filters on the *Y* channel only and find an edge map by thresholding the resulting magnitude of the image gradient appropriately. Which operator gives the best result? Explain and discuss the results.

**b.** Now let's perform edge detection using the Canny edge detector (use the Canny function in MATLAB with appropriate parameters). Compare the result with the edge map found in (a). Explain and discuss the differences.

**c.** Add Gaussian noise to the luminance image (you can use the imnoise function in MATLAB), and find the edges as described in (a) and (b) above. Which method is the best in the presence of noise? Explain why.

**3.4** Enhancement of Color Images: Take two color images. Convert these images from RGB to YCrCb domain.

**a.** Perform the following image enhancement operations on the $Y$ channel only:

**i.** Automatic gain control

**ii.** Histogram equalization

Convert the processed images back to the RGB space for displaying the results. Find the parameters that yield the best results for AGC and unsharp masking. Specify these values with proper justification in your report.

**b.** Implement unsharp masking

**i.** Using simple Gaussian or box low-pass filtering

**ii.** Using bi-lateral filtering instead of low-pass filtering as in Durand and Dorsey [Dur 02]

Compare the results.

**c.** Now apply unsharp masking to the *Y, Cr,* and *Cb* channels independently. Convert the processed images back to the RGB space for displaying the results. Compare the result with the one obtained above. Explain and comment on the differences clearly.

**3.5** Image Denoising (Noise Filtering)

**a.** Add white Gaussian noise with 10 dB SNR to an image to obtain

$$y(n_1, n_2) = s(n_1, n_2) + v(n_1, n_2)$$

**b.** Implement the IIR Wiener filter

$$H_W(k_1, k_2) = \frac{S_{ss}(k_1, k_2)}{S_{ss}(k_1, k_2) + \sigma_v^2}$$

using the 2D-DFT, where the power spectrum of the original image can be estimated by

$$S_{ss}(k_1, k_2) = \frac{1}{N_1 N_2} |Y(k_1, k_2)|^2$$

and the variance of the noise can be estimated as the sample variance of a flat region, such as sky or piece of a flat wall, in the noisy image.

**c.** Implement the adaptive FIR Wiener filter

$$\hat{s}(n_1, n_2) = \hat{\mu}_y(n_1, n_2) + \frac{\hat{\sigma}_s^2(n_1, n_2)}{\hat{\sigma}_y^2(n_1, n_2)} [y(n_1, n_2) - \hat{\mu}_y(n_1, n_2)]$$

where the local sample mean over a $(2M_1 + 1) \times (2M_2 + 1)$ window about $(n_1, n_2)$ can be estimated as

$$\hat{\mu}_y(n_1,n_2) = \frac{1}{(2M_1+1)(2M_2+1)} \sum_{i_1=n_1-M_1}^{n_1+M_1} \sum_{i_2=n_2-M_2}^{n_2+M_2} y(i_1,i_2)$$

$\sigma_y^2(n_1, n_2)$ is estimated by the local sample variance as

$$\hat{\sigma}_y^2(n_1,n_2) = \frac{1}{(2M_1+1)(2M_2+1)} \sum_{i_1=n_1-M_1}^{n_1+M_1} \sum_{i_2=n_2-M_2}^{n_2+M_2} [y(i_1,i_2) - \hat{\mu}_y(n_1,n_2)]^2$$

and the variance of the original image $\sigma_s^2(n_1, n_2)$ can be estimated from the noisy image $y(n_1, n_2)$ as

$$\hat{\sigma}_s^2(n_1,n_2) = \max\left\{0, \hat{\sigma}_y^2(n_1,n_2) - \hat{\sigma}_v^2\right\}$$

**d.** Compare the results of (b) and (c) and write your conclusions.

**3.6** Wavelet Shrinkage

**a.** Add white Gaussian noise with 10 dB SNR to an image to obtain

$$y(n_1, n_2) = s(n_1, n_2) + v(n_1, n_2)$$

**b.** Compute two-level wavelet decomposition of the image $y(n_1, n_2)$ similar to that shown in Figure 3.15 using the MATLAB function *dwt* for 1D-wavelet transform.

**i.** Use Haar wavelet (wname= 'db1')

**ii.** Use an orthogonal wavelet (wname= 'sym8')

**iii.** Use Daubechies 5/3 integer wavelet (wname='rbio3.5')

Display the resulting images. Can you see any differences between the transforms? Comment. (P.S. Write your own code using 1D-wavelet decomposition of first rows and then columns.)

**c.** Apply soft-thresholding in the orthogonal wavelet transform domain and reconstruct the filtered image. Display the filtered image and measure its SNR.

**3.7** Image Restoration

**a.** Apply $5 \times 5$ uniform out-of-focus blur to an image; then add white Gaussian noise with 30 dB SNR to the blurred image to obtain the noisy and blurred image $y(n_1, n_2)$.

b. Compute $|Y(k_1, k_2)|^2$ and display the result as a picture. Can you observe the loci of zero-crossings? How does the picture change if you increase the SNR to 40 dB?

c. Apply the inverse filter (3.97) in the DFT domain to restore the original image. How do you handle image borders? What does the result look like? Explain why.

d. Apply the CLS filter (3.101) in the DFT domain to restore the original image. How do you handle image borders? How does the result change for different choices of $|L(k_1, k_2)|^2$ and different values of $\alpha$? Try at least two different $|L(k_1, k_2)|^2$ and three different values of $\alpha$ for each $|L(k_1, k_2)|^2$.

e. Comment on the tradeoff between noise amplification and regularization artifacts.

## MATLAB Resources

Xin Li, Reproducible Research in Computational Science
http://www.csee.wvu.edu/~xinl/source.html

Xiawen Chen, Fast Bi-lateral Filter and Local Histogram Equalization
http://people.csail.mit.edu/jiawen/#code

Alessandro Foi, BM3D MATLAB Code
http://www.cs.tut.fi/~foi/GCF-BM3D/index.html#ref_software

Peter Kovesi, MATLAB Functions for Computer Vision and Image Processing
http://www.csse.uwa.edu.au/~pk/research/matlabfns/#hysthresh

CHAPTER 4

# Motion Estimation

*Motion estimation refers to estimating 2D image-plane motion (correspondence or optical flow) or 3D motion (object motion or pose). It is a fundamental problem in video processing (motion-compensated filtering/compression) and computer vision.*

Video is a time-varying two-dimensional (2D) spatial-intensity pattern that is formed by projecting a three-dimensional (3D) dynamic scene into a 2D image plane. Temporal variations in the 2D intensity pattern are usually due to relative 3D motion between a camera and objects in the scene. This chapter presents 3D models (in most cases, simplistic ones) for relative motion between a camera and a rigid (static) scene and 2D models for temporal variations of spatial-intensity patterns (pixel motion) in the image plane resulting from such rigid motion as well as independent non-rigid motion of objects in the scene. We start by modeling image formation, which includes different projection models in Section 4.1. Section 4.2 introduces motion models, including 3D rigid motion and 2D pixel motion models. We formulate the 2D pixel motion-estimation problem in Section 4.3. 2D motion-estimation methods can be broadly classified as: i) differential methods that require estimation of spatial- and temporal intensity gradients, ii) search-based matching methods including block-matching and its variations, iii) nonlinear optimization methods including pel-recursive and Bayesian methods, and iv) transform-domain methods, which are discussed

in Section 4.4 to Section 4.7, respectively. Besides being an essential component of image registration/rectification and motion-compensated filtering/compression, 2D-motion estimation is often the first step toward sparse or dense 3D-motion estimation, which is introduced in Section 4.8, from monocular/stereo video.

## 4.1 Image Formation

"Image formation" refers to mapping a 3D scene into an image-plane intensity pattern, which includes geometric projection and photometric effects of motion. Geometric image formation considers camera models for projecting a 3D scene into a 2D image plane, discussed in Section 4.1.1. Photometric image formation that models image intensity variations due to changes in the scene illumination in time as well as the photometric effects of the 3D motion is covered in Section 4.1.2.

### 4.1.1 Camera Models

Imaging systems capture projections of a time-varying 3D scene onto the 2D image plane. Commonly used camera models are *projective camera,* which employs perspective projection, and *affine camera,* which covers orthographic, para-perspective, and weak-perspective projections.

**Projective Camera**

Perspective projection models exact image formation according to the principles of geometrical optics using an ideal pinhole camera, where all rays from the object pass through the center of projection, i.e., the center of the lens. For this reason, it is also known as "central projection." Perspective projection is illustrated in Figure 4.1 for two configurations: a) the image plane $(x_1, x_2)$ coincides with the $(X_1, X_2, 0)$ plane of the scene (world) coordinate system, and the center of projection (lens) is between the object and image planes; and b) the center of projection coincides with the origin of world coordinates.

The relations that describe the perspective projection for the configuration in Figure 4.1(a) are

$$\frac{x_1}{f} = -\frac{X_1}{X_3 - f} \quad \text{and} \quad \frac{x_2}{f} = -\frac{X_2}{X_3 - f}$$

or

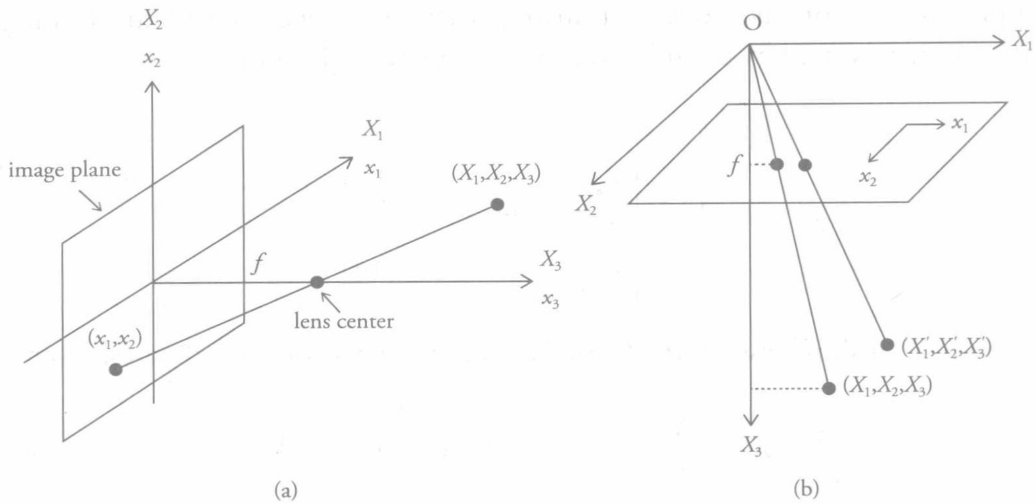



**Figure 4.1** Perspective projection: (a) image plane coincides with the $(X_1, X_2, 0)$ plane of the world coordinate system and (b) center of projection coincides with the origin of the world coordinates.

$$x_1 = \frac{fX_1}{f - X_3} \quad \text{and} \quad x_2 = \frac{fX_2}{f - X_3} \tag{4.1a}$$

where $f$ denotes the focal length of the camera (distance from the center of projection to the image plane), which can be obtained based on the similar triangles formed by drawing perpendicular lines from the object point $(X_1, X_2, X_3)$ and the image point $(x_1, x_2, 0)$ to the $X_3$ axis, respectively.

If we consider a different configuration where the center of projection coincides with the origin of the world coordinates and the image plane is between the object point and camera, which is depicted in Figure 4.1(b), a simple change of variables yields the following equivalent expressions:

$$x_1 = \frac{fX_1}{X_3} \quad \text{and} \quad x_2 = \frac{fX_2}{X_3} \tag{4.1b}$$

The similar triangles used to obtain these expressions are shown in Figure 4.1(b). Observe that the expressions (4.1b) can be employed as an approximate model for the configuration in Figure 4.1(a) when $X_3 \gg f$ with the reversal of the sign because image orientation is the same as that of the object, as opposed to being a mirror image as it should be in actual image formation.

The perspective projection is nonlinear in the Cartesian coordinates since it requires division by the $X_3$ coordinate. However, it can be expressed as a linear

mapping in the homogeneous coordinates, where 3D scene points and 2D image points are represented by 4- and 3-vectors, respectively, given by

$$\mathbf{X}_h = \begin{bmatrix} X_{h,1} \\ X_{h,2} \\ X_{h,3} \\ X_{h,4} \end{bmatrix} \quad \text{and} \quad \mathbf{x}_h = \begin{bmatrix} x_{h,1} \\ x_{h,2} \\ x_{h,3} \end{bmatrix} \tag{4.2}$$

The relationship between the Cartesian and homogeneous coordinates, given by

$$X_1 = \frac{X_{h,1}}{X_{h,4}}, \; X_2 = \frac{X_{h,2}}{X_{h,4}}, \; X_3 = \frac{X_{h,3}}{X_{h,4}} \quad \text{and} \quad x_1 = \frac{x_{h,1}}{x_{h,3}}, \; x_2 = \frac{x_{h,2}}{x_{h,3}} \tag{4.3}$$

is called dehomogenization. Then, expressions (4.1) can be written in a linear form

$$\lambda\, \mathbf{x}_h = \begin{bmatrix} \lambda x_1 \\ \lambda x_2 \\ \lambda \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ 1 \end{bmatrix}$$

where $\lambda$ is a scale parameter (constant). It is not possible to recover the scale parameter without knowing some metric measurement about the scene. This linear relationship can be rewritten as

$$\lambda\, \mathbf{x}_h = \begin{bmatrix} \lambda x_1 \\ \lambda x_2 \\ \lambda \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ 1 \end{bmatrix} \tag{4.4}$$

where the $3 \times 3$ matrix is called the camera calibration (intrinsic parameter) matrix and the $3 \times 4$ matrix is called the extrinsic matrix, which shows the relative positioning of camera and world coordinate systems. In this case, the matrix indicates that the camera and world coordinates are aligned.

The most general form of the perspective projection can be expressed by modifying Eqn. (4.4) to include other intrinsic camera calibration parameters and arbitrary camera pose, i.e., positioning (rotation and translation) of the camera with respect to the (scene) world coordinates, as [Ver 89, Zha 04]

$$\lambda\,\mathbf{x}_h = \begin{bmatrix}\lambda x_1\\ \lambda x_2\\ \lambda\end{bmatrix} = \begin{bmatrix} f k_1 & s & x_{1,0}\\ 0 & f k_2 & x_{2,0}\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1\\ r_{21} & r_{22} & r_{23} & t_2\\ r_{31} & r_{32} & r_{33} & t_3\end{bmatrix}\begin{bmatrix}X_1\\ X_2\\ X_3\\ 1\end{bmatrix}$$

$$= \mathbf{K}\left[\mathbf{R}\,|\,\mathbf{t}\right]\mathbf{X}_h = \mathbf{P}\,\mathbf{X}_h \tag{4.5}$$

where $\mathbf{P} = \mathbf{K}\,[\mathbf{R}\,|\,\mathbf{t}]$ is a $3 \times 4$ projection matrix that models geometric image formation in the most general case. The $3 \times 3$ matrix $\mathbf{K}$ is the camera calibration matrix with 5 degrees of freedom, where $(x_{1,0}, x_{2,0})$ denotes the coordinates of the center of the image, $s = \cot\theta$ is a skewness parameter, where $\theta$ is the angle between $x_1$ and $x_2$ axis, $f$ is the focal length of the camera, and $k_1$, $k_2$ denote the pixel aspect ratio. These parameters are illustrated in Figure 4.2. The $3 \times 3$ matrix $\mathbf{R}$ is a rotation matrix and the $\mathbf{t}$ is a $3 \times 1$ vector, which model the rotation and translation of the camera coordinate system with respect to the scene (world) coordinate system, respectively. Diferent representations for a rotation matrix, with 3 degrees of freedom.

### Affine Camera

Affine camera includes the orthographic, weak-perspective, and paraperspective projection models.

#### *Orthographic Projection*

Ortographic projection is a simple approximation of the actual imaging process where it is assumed that all rays from the 3D object (scene) to the image plane travel

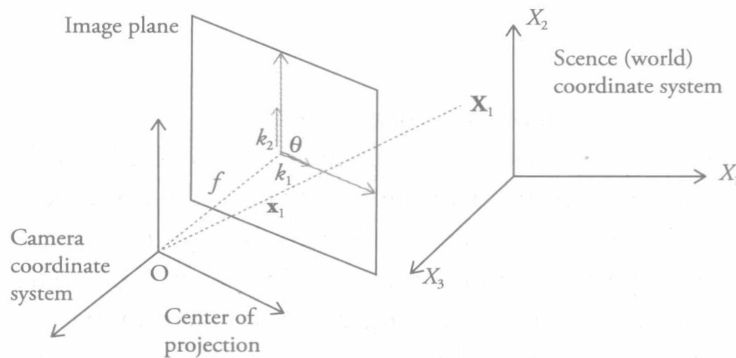



**Figure 4.2** Perspective projection model for arbitrary camera-pose and camera-calibration parameters.

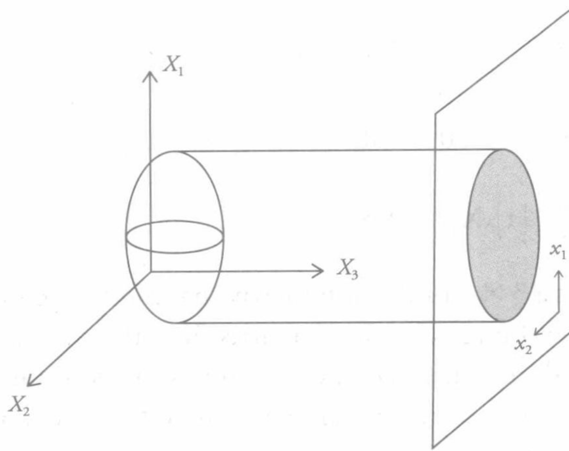



**Figure 4.3** Orthographic projection model.

parallel to each other. Hence, it is sometimes called the "parallel projection," which is illustrated in Figure 4.3, where the image plane is taken parallel to the $(X_1, X_2, 0)$ plane of the world coordinate system.

In this configuration, the orthographic projection can be described in the Cartesian coordinates by

$$x_1 = X_1 \quad \text{and} \quad x_2 = X_2$$

or in vector-matrix notation as

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} \tag{4.6}$$

where $(x_1, x_2)$ denote the image-plane coordinates.

In the orthographic projection, the distance of the object from the camera does not affect image-plane intensity distribution. That is, the object always yields the same size image no matter how far away it is from the camera. Orthographic projection provides a close approximation to the actual image formation when the distance of the object from the camera is much larger than the depth range of points on the object. In such cases, the orthographic projection may be preferred over more complicated but realistic models because it leads to algebraically and computationally more tractable algorithms.

***Weak-Perspective and Paraperspective Projections***

Weak-perspective projection is a scaled-orthographic projection that offers a compromise between the simplicity of the orthographic projection and realism of perspective projection. It is given by

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \frac{f}{Z}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} = \mathbf{M}\,\mathbf{X} \tag{4.7}$$

where $f$ is the focal length of the camera, $Z$ denotes the depth ($X_3$ value along the optical axis) of a reference point in the 3D object, and the $2 \times 3$ matrix $\mathbf{M}$ is called an affine camera matrix.

The paraperspective projection extends scaled-orthographic projection by modeling the perspective deformation of an object with respect to a nearby reference plane that is parallel to the image plane.

## 4.1.2 Photometric Effects of 3D Motion

Image-pixel intensities are modeled proportional to the amount of light reflected by the objects in the scene. The scene reflectance function is generally assumed to contain a Lambertian and a specular component. Here, we assume that the specular component can be neglected. Such surfaces are called Lambertian surfaces, i.e., surfaces whose appearance does not vary with viewpoint. More sophisticated reflectance models can be found in [Lee 90].

### Lambertian Reflectance Model

If a Lambertian surface is illuminated by a single distant-point source with uniform intensity (in time), the resulting image intensity is given by Lambert's cosine law [Lee 90, Pen 91]:

$$s_c(x_1, x_2, t) = \rho\, \mathbf{N}(t) \cdot \mathbf{L} \tag{4.8a}$$

where $\rho$ denotes surface albedo, i.e., the fraction of the light reflected by the surface, $\mathbf{L} = (L_1, L_2, L_3)$ is the unit vector in the mean illuminant direction, and $\mathbf{N}(t)$ is the unit surface normal of the scene, at spatial location $(X_1, X_2, X_3)$ and time $t$, given by

$$\mathrm{N}(t) = \frac{(-\mathrm{p}, -\mathrm{q}, 1)}{\sqrt{\mathrm{p}^2 + \mathrm{q}^2 + 1}} \tag{4.8b}$$

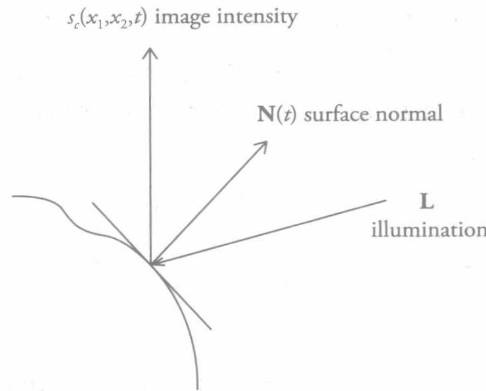



**Figure 4.4** Photometric image-formation model.

in which $p = \delta X_3/\delta x_1$ and $q = \delta X_3/\delta x_2$ are the partial derivatives of depth $X_3(x_1, x_2)$ with respect to the image coordinates $x_1$ and $x_2$, respectively, under the orthographic projection. Photometric image formation is illustrated in Figure 4.4.

The illuminant direction can also be expressed in terms of tilt and slant angles as [Pen 91]

$$\mathbf{L} = (L_1, L_2, L_3) = (\cos\tau \sin\sigma, \sin\tau \sin\sigma, \cos\sigma) \tag{4.9}$$

where $\tau$, the tilt angle of the illuminant, is the angle between $\mathbf{L}$ and $(X_1, X_3)$ plane, and $\sigma$, the slant angle, is the angle between $\mathbf{L}$ and the positive $X_3$ axis.

As a result of relative 3D motion between the scene surface and camera, the surface normal varies by time; so do the photometric properties of the surface. Pentland [Pen 91] shows that the photometric effects of motion can dominate the geometric effects in some cases.

## 4.2 Motion Models

This section discusses modeling of relative motion between a camera and a scene. The 2D image-plane (pixel) motion is the projection of 3D scene motion, which may be due to 3D motion of the camera, change in camera calibration parameters, and/or independent motion of one or more moving objects in the scene. However, 2D "projected motion" may not always be observable from a video (time-varying image intensity) for reasons that are discussed below. Instead, what we observe is the 2D "apparent motion" (optical flow or correspondence). We start by defining

the projected and apparent motion fields and discussing various ambiguities that are inherent in them in Section 4.2.1. Models for the projected motion are presented in Section 4.2.2. 2D apparent-motion models are covered in Section 4.2.3.

### 4.2.1 Projected Motion vs. Apparent Motion

Projected motion refers to the 2D displacement (velocity) field, i.e., projection of the respective 3D vectors into the image plane. In order to estimate the projected motion (together with sparse/dense scene structure) we need sparse/dense correspondence or 2D displacement/optical flow ("apparent motion"), which is estimated from the observed time-varying image intensity (video).

**Projected Motion**

A complete treatment of projected motion models can be found in [Sze 06]. We investigate some cases of common interest, which are often treated under similar problem formulations and solution methods, in the computer-vision and video-processing communities.

*Case 1(a): Camera Motion in a Static Scene or Stereo Vision*

A static scene captured by a moving monocular camera arises in such problems as robot vision for autonomous navigation in static environments and 3D environment modeling by a handheld camcorder, which are often tackled by the computer-vision community. The imaging geometry for this case is depicted in Figure 4.5, where Camera 1 refers to camera position at time $t$ and Camera 2 refers to camera position at time $t'$. This configuration is identical to that of *stereo vision*, where Camera 1 and Camera 2 refer to the left and right cameras, and motion estimation becomes *pose estimation*. In stereo vision, we can lift the requirement that the scene be static, since time is frozen and left and right cameras see the exact same scene even in the presence of independently moving objects.

*Projective Camera* If we have $M$ cameras (multiple frames of video or multi-views) and $N$ sparse feature points that are assumed to be visible in all $M$ frames (views), then the projected image-plane coordinates of the feature points in the homogeneous coordinates are given by

$$\lambda_{ij}\, \mathbf{x}_{ij} = \mathbf{P}_i\, \mathbf{X}_j\; i = 0, \ldots, M-1,\; j = 0, \ldots, N-1 \tag{4.10}$$

where $\mathbf{X}_j$ denotes the $j^{\text{th}}$ feature point in the scene (world) coordinates, $\mathbf{P}_i$ is the $3 \times 4$ projection matrix for the $i^{\text{th}}$ camera, and $\lambda_{\text{ij}}$ are scalars. Let the initial camera

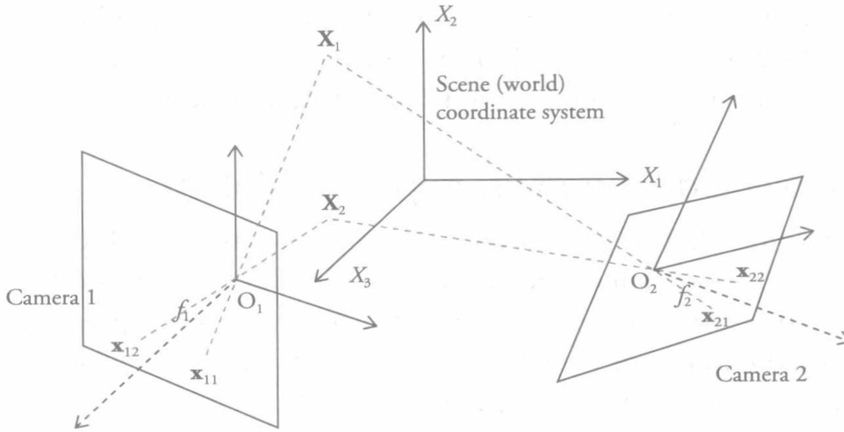



**Figure 4.5** Camera motion with a static scene or stereo vision.

matrix be $\mathbf{P}_0 = \mathbf{K}[\mathbf{I}_{3\times 3} \,|\mathbf{0}_{3\times 1}]$, then the relative pose (motion) between the camera $i$ and camera 0 (the 3D camera coordinates) can be modeled by a rotation matrix $\boldsymbol{R}_i$ and a translation vector $\boldsymbol{T}_i$, $i = 1, \ldots, M-1$.

There is ambiguity in "projected motion" up to an invertible projective transformation $\mathbf{H}$, since

$$\lambda_{ij}\,\mathbf{x}_{ij} = (\mathbf{P}_i\mathbf{H})(\mathbf{H}^{-1}\mathbf{X}_j) = \mathbf{P}_i\mathbf{X}_j \quad i = 1, \ldots, M,\ j = 1, \ldots, N \tag{4.11}$$

Hence, any pair of camera matrices $\mathbf{P}_i\mathbf{H}$ and shape (structure) matrices $\mathbf{H}^{-1}\mathbf{X}_j$ yield the same set of projected image-plane (pixel) coordinates and are projectively equivalent [Har 04].

### *Case 1(b): Rigid Scene Motion with a Static Camera*

If we consider the motion of each object point $\mathbf{X}$ independently, then the 3D displacement vector field can be represented by a set of 3D translation vectors, one for each point. Let a 3D feature point $\mathbf{X}$ at time $t$ move to position $\mathbf{X}'$ at time $t'$ and projections of $\mathbf{X}$ and $\mathbf{X}'$ into the image plane be denoted by pixels $\mathbf{x}$ and $\mathbf{x}'$, respectively. The projection of the 3D motion (scene flow or 3D displacement) vector $\overrightarrow{\mathbf{XX}'}$ into a 2D motion (optical flow or 2D displacement) vector $\overrightarrow{\mathbf{xx}'}$ in the image plane is illustrated in Figure 4.6. When we consider the motion of each object point $\mathbf{X}$ independently, all 3D displacement vectors whose tips lie on the dotted line $\overrightarrow{\mathbf{X}'\mathbf{x}'}$ give the same 2D displacement vector because of the perspective projection.

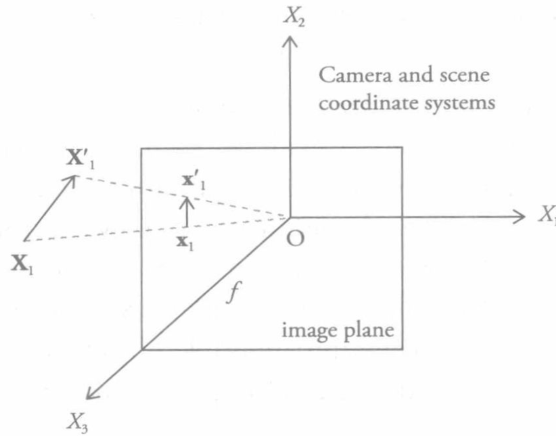



**Figure 4.6** Projection of the 3D scene motion vector onto the image plane.

If the entire field of view consists of a single 3D rigid object or a moving 3D rigid object is segmented from the background, then relative 3D positions of all object points at times $t$ and $t'$ are related by a single rotation matrix $\mathbf{R}$ and a translation vector $\mathbf{T}$, given by

$$\begin{bmatrix} X_1' \\ X_2' \\ X_3' \end{bmatrix} = \mathbf{R} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} + \mathbf{T} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} + \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} \tag{4.12a}$$

which can be expressed in the homogeneous coordinates as

$$\mathbf{X}' = \begin{bmatrix} r_{11} & r_{12} & r_{13} & T_1 \\ r_{21} & r_{22} & r_{23} & T_2 \\ r_{31} & r_{32} & r_{33} & T_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{X} \tag{4.12b}$$

where $\mathbf{X}$ and $\mathbf{X}'$ are the homogeneous coordinates of an object point at times $t$ and $t'$, respectively. We note that this case is equivalent to Case 1(a), where $\mathbf{X}'$ and $\mathbf{X}$ denote the same scene point with respect to two different camera coordinates that are related by Eqn. (4.12).

### *Case 2: Non-Rigid Scene Motion or Multiple Motions with Possible Camera Motion*

This is a typical case in general video processing, such as motion-compensated filtering and compression of generic broadcast TV or surveillance videos. If there are multiple rigid motions or non-rigid 3D motion in the scene, Eqn. (4.10), which assumes a rigid 3D transformation (4.12) between the scene and camera coordinates, is violated. We do not attempt to model non-rigid 3D motion in this book; interested readers should refer to [Ter 88, Tan 94]. This case may be addressed by segmenting the scene into multiple objects, each exhibiting a single rigid 3D motion or directly modeling resulting 2D "apparent" motion (discussed in Section 4.2.3).

### Apparent Motion – Optical Flow

Apparent motion refers to correspondence (displacement) or optical flow (velocity) field that is perceived (can be observed) from the time-varying image intensity pattern (video). We observe that "apparent motion" (correspondence or optical flow) is, in general, different from the "projected motion" (displacement or velocity) field due to following ambiguities [Ver 89]:

- Lack of sufficient spatial-image gradient: There must be sufficient gray-level (color) variation within moving regions for the actual motion to be observable. An example of an unobservable motion is shown in Figure 4.7, where a circle with uniform intensity rotates about its center. This motion generates no optical flow, and thus is unobservable.

  We often perform motion estimation over small blocks of pixels, called a finite aperture. When this finite aperture does not contain sufficient image gradient, the motion is not observable within that aperture, which is referred as the aperture problem (see Section 4.3.4).

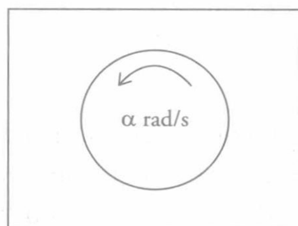



**Figure 4.7** All projected motion does not generate optical flow.

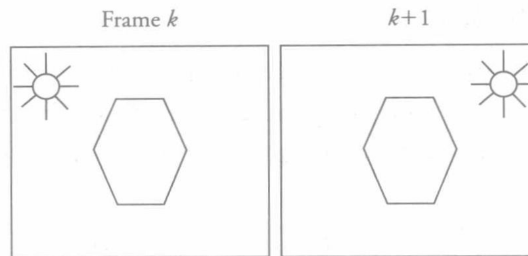



**Figure 4.8** All optical flow does not correspond to projected motion.

- Changes in external illumination and or shading: An observable optical flow may not always correspond to an actual motion. For example, if the intensity and/or direction of external illumination vary from frame to frame, as shown in Figure 4.8, then an optical flow will be observed even though there is no motion. Therefore, changes in the external illumination impair estimation of the actual 2D motion field if it is not properly modeled.

  In some cases, the shading may vary from frame to frame due to 3D motion of objects even if there is no change in the external illumination. For example, if an object rotates, its surface normal changes, which results in a change in the shading. This change in shading, discussed in Section 4.1.2, may cause the intensity of the pixels along a motion trajectory to vary, which needs to be taken into account.

Because 3D and 2D motion estimation problems are both ill-posed due to various ambiguities that are discussed, motion estimation methods need additional assumptions (models) about the structure of the 2D motion field for regularization of the problem, which are discussed next. General discussion of deterministic smoothness models and Markov random field models can be found in Appendices B and C, respectively.

## 4.2.2 Projected 3D Rigid-Motion Models

According to classical kinematics, 3D motion can be classified as rigid vs. non-rigid motion. In rigid motion, relative distances between a set of 3D points remain fixed as the scene evolves in time. This section presents exact models to describe the projection of relative rigid 3D motion of a set of object points and a camera, which can be derived for Cases 1(a) and 1(b) (discussed earlier).

**General Model (Depth Map)**

If we make no assumptions about the scene structure and take the depth of each feature point $X_3$ as an independent variable, then projecting the 3D feature point position $\mathbf{X}'$ given by (4.12) into the image plane using (4.1), we have

$$x_1' = \frac{fX_1'}{X_3'} = \frac{f(r_{11}X_1 + r_{12}X_2 + r_{13}X_3 + T_1)}{r_{31}X_1 + r_{32}X_2 + r_{33}X_3 + T_3}$$
$$x_2' = \frac{fX_2'}{X_3'} = \frac{f(r_{21}X_1 + r_{22}X_2 + r_{23}X_3 + T_2)}{r_{31}X_1 + r_{32}X_2 + r_{33}X_3 + T_3}$$

Dividing the numerator and denominator of both expressions by $X_3$, we have

$$x_1' = \frac{r_{11}x_1 + r_{12}x_2 + f r_{13} + {}^{fT_1}\!/_{X_3}}{r_{31}x_1 + r_{32}x_2 + r_{33} + {}^{T_3}\!/_{X_3}} \tag{4.13a}$$

$$x_2' = \frac{r_{21}x_1 + r_{22}x_2 + f r_{23} + {}^{fT_2}\!/_{X_3}}{r_{31}x_1 + r_{32}x_2 + r_{33} + {}^{T_3}\!/_{X_3}} \tag{4.13b}$$

where the depth $X_3$ of each feature point appears as a parameter in these expressions. Here, the six 3D motion parameters (three rotation and three translation) constrain the direction of 2D image motion (displacement or flow) vectors, while the depth parameter is required to determine the exact value of the 2D motion vector.

**Homography (Perspective Model)**

If the 3D structure (shape) of a moving object can be modeled by a non-deformable surface, e.g., a planar or piecewise planar surface, to relate the depth of object points, then the number of free depth parameters can be reduced. Let the set of 3D feature points lie on a plane, described by

$$aX_1 + bX_2 + cX_3 = 1$$

where $[a \quad b \quad c]^T$ denotes the normal vector of the plane. Since the right-hand-side is unity, the 3D displacement model (4.4) can be rewritten as

$$\begin{bmatrix} X_1' \\ X_2' \\ X_3' \end{bmatrix} = \mathbf{R} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} + \mathbf{T} = \mathbf{R} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} + \mathbf{T} \left( [a \quad b \quad c] \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} \right) = \mathbf{H} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix}$$

where $\mathbf{H} = \mathbf{R} + \mathbf{T}[a \quad b \quad c]$. Now, projecting 3D scene coordinates into the 2D image plane, following steps similar to those used to obtain (4.5), we have the homography relations, which model perspective projection of 3D motion of a planar surface, given by

$$x_1' = \frac{h_1 x_1 + h_2 x_2 + h_3}{h_7 x_1 + h_8 x_2 + h_9} \tag{4.14a}$$

$$x_2' = \frac{h_4 x_1 + h_5 x_2 + h_6}{h_7 x_1 + h_8 x_2 + h_9} \tag{4.14b}$$

where $h_9$ is sometimes set equal to 1 in order to account for the scale ambiguity. Eqn. (4.14) is also known as the perspective model.

We note that if $\mathbf{T} = \mathbf{0}$, we have $\mathbf{H} = \mathbf{R}$ and the homography (4.14) provides an exact mapping between two image frames (i.e., compensating for camera calibration and rotation) regardless of the scene geometry. In summary, two frames are related by a homography if and only if

1. they are views of the same 3D planar surface from different camera positions (with rotation and translation).
2. they are captured by the same camera where the camera is only allowed to rotate about its optical center and/or zoom (without any translation). This case is independent of the scene structure.

If there are multiple planar objects with different motions or when the surface of a single moving object is approximated by a piecewise planar model, then a different parameter set, $h_1, \ldots, h_8$, is required to describe the motion of pixels for each planar piece, which is often called a layered scene representation (see Chapter 5).

### Residual Planar-Parallax Motion Model

While the general model (4.5) measures feature point depth $X_3$ with respect to the camera coordinates, the depth can also be measured with respect to a reference plane

in the scene. A reference plane that is visible in both frames can be aligned using the homography given by (4.14), which compensates motion due to change in camera calibration and camera rotation. Any residual motion is due only to camera translation and deviation of the scene structure from the reference plane. This residual image-plane motion, called planar parallax motion, is a radial vector field centered at the epipole (focus of expansion) $\mathbf{e} = [e_1\ e_2\ e_3]^T$ in the homogeneous coordinates.

The image-plane displacement in homogeneous coordinates can be decomposed as

$$\mathbf{x}' - \mathbf{x} = (\mathbf{x}' - \mathbf{x}_w) + (\mathbf{x}_w - \mathbf{x})$$

where $\mathbf{x}_w$ is obtained by warping $\mathbf{x}$ to the coordinate system of the second camera using the homography (4.14). Here, $\mathbf{x}' - \mathbf{x}_w$ represents the planar-motion component. The residual (planar parallax) motion $\mathbf{x}_w - \mathbf{x}$ can be modeled by

$$\mathbf{x}_w - \mathbf{x} = -\frac{\gamma}{1 + \gamma e_3}(e_3 \mathbf{x} - \boldsymbol{e}) \tag{4.15}$$

where $\gamma = \dfrac{H}{X_3}$ and $H$ denotes the vertical distance of the 3D point $\mathbf{X}$ from the reference plane in the scene, which is a shape-invariant feature. The derivation of (4.15) can be found in [Ira 02, Ira 98].

### 4.2.3 2D Apparent-Motion Models

This section introduces parametric and non-parametric 2D apparent-motion models that are either approximations to the projected motion model or aim to impose a local smoothness constraint to regularize apparent motion estimation.

#### Parametric Models

Parametric models aim to describe 2D apparent motion (displacement or flow) of a video frame or a block of pixels with a small number of parameters. Assuming there is no occlusion, the $k+1^{\text{st}}$ frame of a sequence can be expressed as $\mathbf{s}_{k+1}(\mathbf{x}) = \mathbf{s}_k(\mathbf{x}')$, where $\mathbf{x}' = h(\mathbf{x}; \Theta)$ is a transformation of pixels from frame $k$ to $k+1$, given the parameter vector $\Theta$. The transformation $h(\mathbf{x}; \Theta)$ must be unique and invertible. Homography, discussed in Section 4.2.2, is an example for parametric models with 8 degrees of freedom (free parameters); however, it is nonlinear in the parameters since it involves division. We introduce simpler models and linear approximations to homography in this section.

### *Block-Translation Model*

The simplest motion model is block translation, which assumes a frame is composed of moving blocks, whose motion can be characterized by a translation vector $\mathbf{d} = (d_1, d_2)^T$, i.e.,

$$x_1' = x_1 + d_1 \tag{4.16a}$$

$$x_2' = x_2 + d_2 \tag{4.16b}$$

where $(x_1', x_2')$ denotes coordinates of pixel $(x_1, x_2)$ in the reference frame $k \mp l$. It is used in video-compression standards, since it is simple and effective for modeling small motions that can be approximated by translation. Two motion-estimation methods specifically designed for estimating block translation are block-matching and phase-correlation, discussed in Sections 4.5 and 4.7, respectively.

### *Affine Model*

The orthographic projection of 3D rigid motion of a planar surface can be described by a six-parameter affine model. It can be applied to an image frame or just a block of pixels, given by

$$x_1' = a_1 x_1 + a_2 x_2 + a_3 \tag{4.17a}$$

$$x_2' = a_4 x_1 + a_5 x_2 + a_6 \tag{4.17b}$$

It provides a good approximation to homography if the distance of the planar surface from the camera is large enough so that all rays from the planar object to the camera can be assumed parallel. Special cases of the affine model include the following 2D image (pixel) motions:

1. Pure translation: If $a_2 = a_4 = 0$ and $a_1 = a_5 = 1$, then Eqn. (4.17) reduces to (4.16), with $a_3 = d_1$ and $a_6 = d_2$.
2. Pure rotation: If $a_3 = a_6 = 0$, $a_1 = \cos\theta$, $a_2 = \sin\theta$, $a_4 = -\sin\theta$, and $a_5 = \cos\theta$, then

$$x_1' = x_1 \cos\theta + x_2 \sin\theta$$
$$x_2' = -x_1 \sin\theta + x_2 \cos\theta$$

   models rotation of the axis in the image plane by angle $\theta$.

3. *Isometry* refers to combination of rotation and translation with 3 degrees of freedom, $\theta, a_3, a_6$.
4. Isotropic scaling (Zoom): If $a_1 = a_5 = k$ and $a_2 = a_3 = a_4 = a_6 = 0$, then

$$x_1' = k\, x_1$$
$$x_2' = k\, x_2$$

5. *Similarity transformation* refers to a combination of rotation, translation, and scaling:

$$x_1' = kx_1 \cos\theta + x_2 \sin\theta + d_1$$
$$x_2' = -x_1 \sin\theta + kx_2 \cos\theta + d_2$$

In its most general form, the afne transformation (4.17) preserves parallel lines in the image plane, i.e., two parallel lines in frame $k$ are mapped to parallel lines in frame $k+1$.

*Linear Approximations to the Perspective-Motion Model*

The bi-quadratic, pseudo-perspective, and bilinear models are linear approximations to homography (perspective model). The bi-quadratic model, with 12 free parameters,

$$x_1' = a_1x_1 + a_2x_2 + a_3 + a_4x_1^2 + a_5x_2^2 + a_6x_1x_2 \tag{4.18a}$$

$$x_2' = a_7x_1 + a_8x_2 + a_9 + a_{10}x_1^2 + a_{11}x_2^2 + a_{12}x_1x_2 \tag{4.18b}$$

can be obtained by a Taylor series expansion of the homography. The pseudo-perspective model, with eight free parameters, given by

$$x_1' = a_1x_1 + a_2x_2 + a_3 + a_7x_1^2 + a_8x_1x_2 \tag{4.19a}$$

$$x_2' = a_4x_1 + a_5x_2 + a_6 + a_7x_1x_2 + a_8x_2^2 \tag{4.19b}$$

is an instantaneous flow approximation. The bilinear (pseudo-perspective) model, given by

$$x_1' = a_1x_1 + a_2x_2 + a_3x_1x_2 + a_4 \tag{4.20a}$$

$$x_2' = a_5 x_1 + a_6 x_2 + a_7 x_1 x_2 + a_8 \tag{4.20b}$$

is another linearized approximation. Parameters of these models can be estimated directly from image intensity (Section 4.4.1) or from given/pre-computed feature correspondences (Section 4.5.5).

### Non-Parametric Models

Unlike parametric models, non-parametric models can be used to estimate 2D motion, which is projection of non-rigid and deformable 3D motion. Non-parametric models impose smoothness constraints on the estimated 2D-motion field without any assumptions about the nature of the underlying 3D motion and scene structure. Use of smoothness constraints to regularize solutions of ill-posed problems is well-known in science and engineering [Ber 88] (see Appendix A). Te non-parametric models can be classified as deterministic vs. stochastic models.

#### *Deterministic Models*

Deterministic models impose a smoothness constraint on the 2D-motion field, which requires that motion vectors vary slowly from pixel to pixel or block to block over a spatio-temporal neighborhood. They may take several forms: i) In differential methods, imposing a global or local smoothness constraint on the solution of the optical flow equation requires solution of a variational problem [Hor 81] (Section 4.4). Because a global smoothness constraint causes inaccurate motion estimation at motion/occlusion boundaries, more advanced directional smoothness constraints that allow for sudden discontinuities in the motion field have also been proposed [Nag 86]. ii) In the block-matching method (Section 4.5), commonly used in video-compression standards, the search can be initialized at a pixel pointed by the estimate from one of neighboring blocks. iii) Pel-recursive methods are predictor-corrector type displacement estimators (Section 4.6), where the prediction at each pixel can be taken as the motion estimate at the previous pixel or as a linear combination of estimates in a neighborhood of the current pixel. Hence, the prediction step can be considered as an implicit smoothness constraint. Wiener-type estimation extends this concept to block-based recursive-motion estimation.

#### *Probabilistic Smoothness Constraints*

Bayesian motion-estimation methods utilize probabilistic smoothness constraints, usually in the form of a Gibbs random field, where smoothness of the displacement feld is quantifed in terms of some energy functions (see Appendix B). It is also possible to impose directional smoothness constraints within this framework by

defining line fields [Dub 93]. The main drawback of such methods is the extensive amount of computation that is required.

## 4.3 2D Apparent-Motion Estimation

While it is the projected "true" motion that is desired in most computer-vision problems, estimation of apparent motion is often sufficient for most video-processing applications such as motion-compensated filtering and video compression. This section first defines various formulations of the 2D apparent-motion estimation problem and clarifies the distinction between them in Section 4.3.1. Motion estimation from one or more frames relies on the principle (assumption) that image intensity remains constant along the true-motion path (trajectory). This constraint is mathematically stated in the form of the optical flow equation (OFE) in Section 4.3.2 and in the form of displaced frame difference (DFD) in Section 4.3.3. Section 4.3.4 discusses ambiguities in pixel-based estimation of 2D apparent motion. The concept of hierarchical-motion estimation is introduced in Section 4.3.5. Finally, Section 4.3.6 presents measures to assess motion estimation performance.

### 4.3.1 Sparse Correspondence, Optical-Flow Estimation, and Image-Registration Problems

The 2D "apparent motion" estimation problem can be posed as sparse correspondence estimation or dense displacement/velocity (optical flow) estimation or global image registration problem.

#### Sparse-Correspondence Estimation

The displacement of an image feature or pixel from image coordinate $\mathbf{x}$ at time $t$ to $\mathbf{x}'$ at time $t'$ results in a displacement (correspondence) vector $\mathbf{d}(\mathbf{x}, t) = \mathbf{x}' - \mathbf{x}$. The sparse-correspondence estimation problem can be posed as finding the displacement vectors $\mathbf{d}(\mathbf{x}_j, t)$ between two or more frames at some pre-determined isolated "good" feature points $\mathbf{x}_j$, $j = 1, \ldots, N$. Good feature points are typically corner points or pixels of interest that can be uniquely matched in two or more frames. That is, there is a sufficient image gradient in their neighborhood and they are visible in the frames of interest (see aperture and occlusion problems in Section 4.3.4). Detection of "good" feature points, with sufficient image gradient in their local neighborhood, has been discussed in Section 3.3.4. Sparse feature correspondence estimation is usually the first step in 3D motion and structure estimation (see Section 4.8).

### Dense-Motion (Optical-Flow/Displacement) Estimation

Apparent flow of image intensity pattern on a lattice of pixels $(\mathbf{x}, t) \in \Lambda^3$ is called optical flow. Optical flow field is the set of instantaneous velocity vectors $\mathbf{v}(\mathbf{x}, t) = (v_1(\mathbf{x}, t), v_2(\mathbf{x}, t))^T = [dx_1/dt \ dx_2/dt]^T$ for all $(\mathbf{x}, t) \in \Lambda^3$. The dense correspondence (displacement) field is the set of 2D displacement vectors $\mathbf{d}(\mathbf{x}, t)$, for all $(\mathbf{x}, t) \in \Lambda^3$. The correspondence and optical flow vectors usually vary from pixel to pixel (space-varying motion), e.g., due to rotation of objects.

The 2D dense-motion estimation problem can be posed as estimation of either:

1. correspondence vectors $\mathbf{d}(\mathbf{x}, t) = [d_1(\mathbf{x}, t), d_2(\mathbf{x}, t)]^T$ for all $(\mathbf{x}, t) \in \Lambda^3$ or
2. optical-flow vectors $\mathbf{v}(\mathbf{x}, t) = [v_1(\mathbf{x}, t), v_2(\mathbf{x}, t)]^T$ for all $(\mathbf{x}, t) \in \Lambda^3$.

The lattice $\Lambda^3$ may consist of all pixels or a subset of them (as in block-based motion estimation used in video-compression standards).

#### *Dense-Correspondence Problem*

The dense-correspondence problem can be set up as a *forward-* or *backward-*motion estimation problem, as depicted in Figure 4.9, depending on whether the motion vector is defined from $t$ to $t + l\Delta t$ or from $t$ to $t - l\Delta t$, where $l$ is the frame counter and $\Delta t$ is the temporal sampling interval.

***Forward Estimation*** Given two video frames at $t$ and $t + l\Delta t$ that are related by

$$s_p(x_1, x_2, t) = s_c(x_1 + d_1(\mathbf{x}), x_2 + d_2(\mathbf{x}), t + l\Delta t) \tag{4.21}$$

where the temporal argument of $\mathbf{d}(\mathbf{x})$ is dropped for ease of notation, or equivalently,

$$s_k(x_1, x_2) = s_{k+l}(x_1 + d_1(\mathbf{x}), x_2 + d_2(\mathbf{x}))$$

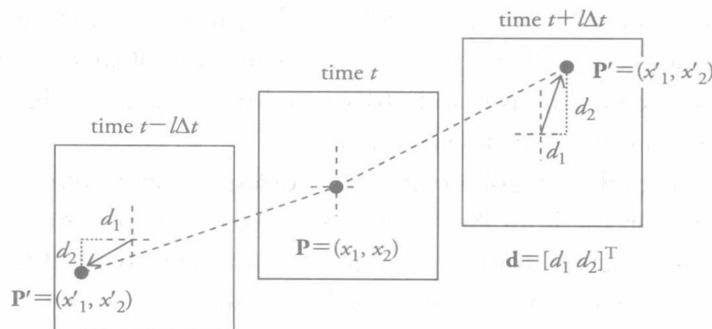



**Figure 4.9** Forward- and backward-correspondence estimation.

such that $t = k\Delta t$, find correspondence vectors $\mathbf{d}(\mathbf{x}) = [d_1(\mathbf{x}), d_2(\mathbf{x})]^T$ for all $(\mathbf{x}, t) \in \Lambda^3$.

***Backward Estimation*** If we define correspondence vectors from time $t$ to $t - l\Delta t$, then

$$s_k(x_1, x_2) = s_{k-l}(x_1 + d_1(\mathbf{x}), x_2 + d_2(\mathbf{x}))$$

such that $t = k\Delta t$. Alternately, the motion vector can be defined from time $t - l\Delta t$ to $t$. Then,

$$s_k(x_1, x_2) = s_{k-l}(x_1 - d_1(\mathbf{x}), x_2 - d_2(\mathbf{x}))$$

In predictive video compression, backward-motion estimation is used in forward (P-mode) causal-motion compensation. Because $\mathbf{x} \pm \mathbf{d}(\mathbf{x})$ does not generally correspond to a lattice site (integer pixel), the right-hand sides of these expressions must be evaluated using some sub-pixel interpolation scheme. The dense correspondence problem also arises in stereo-disparity estimation, where we have a left-right pair instead of a temporal pair of images.

*Optical-Flow Estimation Problem*

The optical-flow estimation problem can be posed as: given samples of $s_p(x_1, x_2, t)$ on a 3D lattice $\Lambda^3$, determine the 2D instantaneous velocity $\mathbf{v}(\mathbf{x}, t)$ for all pixels $(\mathbf{x}, t) \in \Lambda^3$.

Theoretically, continuous spatio-temporal intensity pattern $s_c(\mathbf{x}, t)$ is required to determine the optical-flow field, since we need to analyze spatial and temporal variations in the continuous spatio-temporal intensity pattern by computing partial derivatives. However, in practice, we estimate the optical-flow field from the available video data, which is spatially and temporally discrete intensity on a spatio-temporal lattice $\Lambda^3$. Therefore, accurate estimation of spatial/temporal partial derivatives from discrete intensity data (discussed in Chapter 3) plays an important role in the precision of optical-flow estimates.

We can make the following observations: i) correspondence vectors converge to the optical-flow vectors in the limit $\Delta t = t' - t$ goes to zero; ii) estimation of optical flow and correspondence vectors from two frames are equivalent, with $\mathbf{d}(\mathbf{x}, t) = \mathbf{v}(\mathbf{x}, t)\ l\Delta t$, provided that the velocity remains constant during each time interval $l\Delta t$; and iii) we need to consider more than two frames at a time to estimate optical flow in the presence of acceleration.

**Image-Registration Problem**

Image registration is a special case of the correspondence problem, where a global parametric mapping or a collection of local parametric mappings can be defined between pairs of frames to register them on a common reference frame. For example, multiple exposures of a static scene taken by a panning camera can be registered by one of the parametric models in Section 4.2.3.

*Mosaic Representation (Image Stitching)*

Because individual images in a camera-pan sequence have varying field of view with some overlap between them, a panoramic field of view can be obtained by stitching them together with proper blending of intensity of pixels that are registered on a single reference frame called a photo-mosaic [Ira 96, Sze 06]. Most recent digital cameras provide this functionality as a single-button option.

## 4.3.2 Optical-Flow Equation and Normal Flow

The fundamental principle of motion estimation is that the intensity of a pixel remains constant along the motion trajectory, which can be expressed in the form of the OFE. Assuming that space and time are represented by continuous variables, intensity constancy implies that the rate of change of intensity along the motion trajectory is zero, expressed as

$$\frac{d\, s_c(x_1, x_2, t)}{dt} = 0 \tag{4.22}$$

This is a total derivative expression since $x_1$ and $x_2$ vary with $t$ along the motion trajectory. Using the chain rule of differentiation, we have

$$\frac{\partial s_c(x_1, x_2, t)}{\partial x_1}\frac{\partial x_1}{\partial t} + \frac{\partial s_c(x_1, x_2, t)}{\partial x_2}\frac{\partial x_2}{\partial t} + \frac{\partial s_c(x_1, x_2, t)}{\partial t} = 0 \tag{4.23}$$

where $v_1(\mathbf{x}, t) = \partial x_1/\partial t$ and $v_2(\mathbf{x}, t) = \partial x_2/\partial t$ denote the components of the coordinate velocity vector in terms of the continuous spatial coordinates. This is known as the OFE or the optical-flow constraint, which can alternatively be expressed in vector form as

$$\langle \nabla s_c(\mathbf{x}, t), \mathbf{v}(\mathbf{x}) \rangle + \frac{\partial s_c(\mathbf{x}, t)}{\delta t} = 0 \tag{4.24}$$

where $\nabla s_c(\mathbf{x},t) = \left[\frac{\partial s_c(\mathbf{x},t)}{\delta x_1}, \frac{\partial s_c(\mathbf{x},t)}{\delta x_2}\right]^T$ and $\langle \cdot,\cdot \rangle$ denotes the vector inner product.

**Can the OFE Uniquely Specify the Motion Field?**

The OFE (4.24) is not sufficient to uniquely specify the 2D velocity (flow) field, since it yields one scalar equation in two unknowns, $v_1(\mathbf{x}, t)$ and $v_2(\mathbf{x}, t)$, at each pixel site $(\mathbf{x}, t)$. Inspection of (4.24) reveals that we can only estimate the component of the flow vector that is in the direction of the spatial-image gradient $\nabla s_c(\mathbf{x}, t)$, called the normal flow $\mathbf{v}_\perp(\mathbf{x}, t)$, because the component that is orthogonal to the spatial-image gradient disappears under the dot product. The concept of normal flow is illustrated in Figure 4.10, where all vectors whose tip lie on the dotted line satisfy Eqn. (4.24). The optical flow equation (4.24) can be rewritten as

$$\| \nabla s_c(\mathbf{x},t) \| \| \mathbf{v}(\mathbf{x},t) \| \cos\beta + \frac{\partial s_c(\mathbf{x},t)}{\delta t} = 0$$

where $\beta$ is the angle between the vectors $\nabla s_c(\mathbf{x}, t)$ and $\mathbf{v}(\mathbf{x}, t)$. Then, the magnitude of normal flow $\|\mathbf{v}_\perp(\mathbf{x}, t)\|$ at each site can be computed by setting the angle $\beta = 0$

$$\| \mathbf{v}_\perp(\mathbf{x},t) \| = \frac{-\frac{\partial s_c(\mathbf{x},t)}{\delta t}}{\| \nabla s_c(\mathbf{x},t) \|} \tag{4.25}$$

Thus, without additional motion-field modeling or assumptions, we can only determine motion that is parallel to the spatial-image gradient vector (orthogonal to

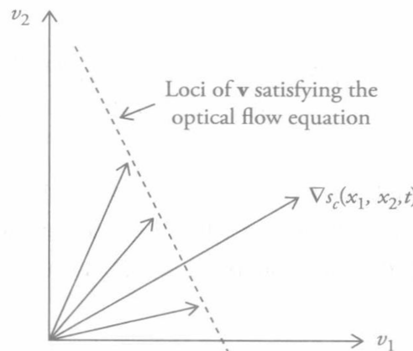



**Figure 4.10** Normal flow. All vectors whose tip lie on the dotted line satisfy (4.24).

the edge), called the normal flow, at each pixel. Observe that the OFE requires that i) the spatio-temporal image intensity be differentiable, and ii) the partial derivatives of the intensity be estimated.

### 4.3.3 Displaced-Frame Difference

The displaced-frame difference (DFD) equation expresses the main principle of motion estimation that the intensity of a pixel remains constant along the motion trajectory in discrete spatial and temporal notation. In the case of forward-motion estimation, the DFD between time instances $t$ and $t' = t + \Delta t$ is defined by

$$DFD(\mathbf{x},\mathbf{d}) = s_c(\mathbf{x}+\mathbf{d}(\mathbf{x}), t+\Delta t) - s_c(\mathbf{x},t) \tag{4.26}$$

where $s_c(\mathbf{x}, t)$ is the video and $\mathbf{d}(\mathbf{x}) = [d_1(\mathbf{x}), d_2(\mathbf{x})]^T$ denotes the motion vector (MV) field between times $t$ and $t + \Delta t$. We observe that i) since the components of $\mathbf{d}(\mathbf{x})$ are allowed to take non-integer values, interpolation is required to compute the DFD, and ii) if $\mathbf{d}(\mathbf{x})$ is equal to the true MV and there is no interpolation error, the DFD attains the value zero.

We can expand $s_c(\mathbf{x} + \mathbf{d}(\mathbf{x}), t + \Delta t)$ into a Taylor series about $(\mathbf{x}, t)$, for small $\mathbf{d}(\mathbf{x})$ and $\Delta t$, as

$$\begin{aligned} s_c(x_1 + d_1(\mathbf{x}), x_2 + d_2(\mathbf{x}), t+\Delta t) &= s_c(\mathbf{x},t) + \\ d_1(\mathbf{x})\frac{\partial s_c(\mathbf{x},t)}{\partial x_1} &+ d_2(\mathbf{x})\frac{\partial s_c(\mathbf{x},t)}{\partial x_2} + \Delta t \frac{\partial s_c(\mathbf{x},t)}{\partial t} + \ h.o.t. \end{aligned} \tag{4.27}$$

Substituting (4.27) into (4.26), and neglecting the higher-order terms (h.o.t.),

$$DFD(\mathbf{x},\mathbf{d}) = \frac{\partial s_c(x_1,x_2,t)}{\partial x_1} d_1(\mathbf{x}) + \frac{\partial s_c(x_1,x_2,t)}{\partial x_2} d_2(\mathbf{x}) + \frac{\partial s_c(x_1,x_2,t)}{\partial t} \Delta t \tag{4.28}$$

We investigate the relationship between the DFD and OFE in two cases.

**Case 1**

Limit $\Delta t \to 0$: Setting $DFD(\mathbf{x}, \mathbf{d}) = 0$, dividing both sides of (4.28) by $\Delta t$, and taking the limit as $\Delta t$ approaches 0, we obtain the OFE

$$\frac{\partial s_c(x_1,x_2,t)}{\partial x_1} v_1(\mathbf{x},t) + \frac{\partial s_c(x_1,x_2,t)}{\partial x_2} v_2(\mathbf{x},\mathrm{t}) + \frac{\partial s_c(x_1,x_2,t)}{\partial t} = 0$$

where $\mathbf{v}(\mathbf{x}, t) = [v_1(\mathbf{x}, t), v_2(\mathbf{x}, t)]^T$ denotes the velocity vector at time $t$. We see that setting the DFD equal to zero (or minimizing the DFD) is equivalent to imposing the OFE (4.24) in the limit as $\Delta t \rightarrow 0$.

Case 2

For $\Delta t$ finite: An estimate of the displacement vector $\hat{\mathbf{d}}(\mathbf{x})$ between any two frames that are $\Delta t$ apart can be obtained from (4.26) in a number of ways:

1. Search for $\hat{\mathbf{d}}(\mathbf{x})$, which would set the left-hand side of Eqn. (4.26) to zero over a block of pixels (block-matching strategy).
2. Compute $\hat{\mathbf{d}}(\mathbf{x})$, which would set the left-hand side of (4.26) or (4.28) to zero on a pixel-by-pixel basis using a gradient-based optimization scheme (pel-recursive strategy).
3. Set $\Delta t = 1$ and $DFD(\mathbf{x}, \hat{\mathbf{d}}(\mathbf{x})) = 0$; solve for $\hat{\mathbf{d}}(\mathbf{x})$ using a set of linear equations obtained from the right-hand side of (4.28) using a block of pixels.

All three approaches can be shown to be identical if i) local variation of the spatio-temporal image intensity is linear, and ii) velocity is constant within the time interval $\Delta t$, i.e.,

$$\hat{d}_1(\mathbf{x}) = \hat{v}_1(\mathbf{x}, t)\Delta t \quad \text{and} \quad \hat{d}_2(\mathbf{x}) = \hat{v}_2(\mathbf{x}, t)\Delta t$$

In practice, $DFD(\mathbf{x}, \mathbf{d})$ hardly ever becomes exactly zero for any value of $\mathbf{d}$, because: i) there is observation noise, ii) there is occlusion (covered/uncovered regions), iii) there are interpolation errors for non-integer MV, and iv) scene illumination may vary frame to frame. Therefore, we minimize the absolute value or square of the DFD or the left-hand side of the OFE over a block of pixels to estimate the 2D-motion field. Pel-recursive methods (see Section 4.6) employ gradient-based optimization to minimize the square of the DFD with an implicit smoothness constraint (as opposed to search methods used in block-matching).

### 4.3.4 Motion Estimation is Ill-Posed: Occlusion and Aperture Problems

2D-motion estimation, posed as either a correspondence or optical-flow estimation problem, based on two frames, is an "ill-posed" problem in the absence of any additional assumptions about the nature of the motion. A problem is called ill-posed if a unique solution does not exist, and/or solution(s) do(es) not continuously depend

on the data [Ber 88]. 2D-motion estimation suffers from all of existence, uniqueness, and continuity problems:

- Existence of a solution: No correspondence can be established for covered/uncovered background points. This is known as the *occlusion* problem.
- Uniqueness of the solution: Treating $x_1$ and $x_2$ components of displacement (or velocity) at each pixel as independent variables, the number of unknowns is twice the number of equations (the frame difference at each pixel). Hence, the motion-estimation problem is under-determined.
- Continuity of the solution: A small amount of observation noise in video frames may result in a large deviation in the motion estimates, i.e., the estimate is highly sensitive to noise.

Therefore, the motion-estimation problem must be regularized by using motion models and/or priors.

### Occlusion Problem

Occlusion refers to covering/uncovering part of an object or background from frame to frame due to motion of an object with respect to the camera, such that some pixels in the current frame do not have a correspondence in the reference frame. There are two sources of occlusion:

1. mutual occlusion, where a moving object covers another object or part of the background (see Figure 4.11) and
2. self-occlusion, when for example, an object rotates clockwise out of plane, the left edge gets covered and some new texture is uncovered from the right edge.

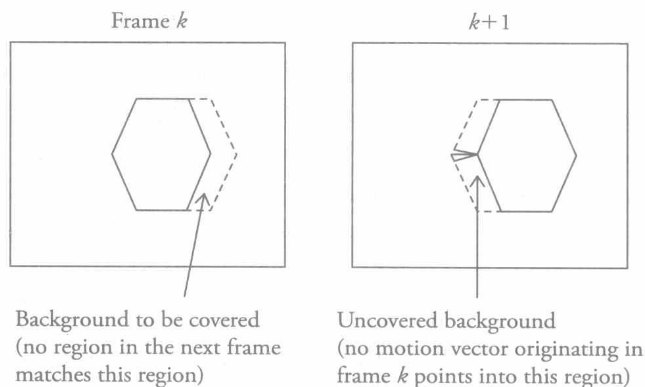



**Figure 4.11** Covered/uncovered background problem.

Covered and uncovered background concepts are illustrated in Figure 4.11, where an object shown by solid lines translates in the $x_1$ direction from frames $k$ to $k+1$. The dotted region in frame $k$ indicates the background to be covered in frame $k+1$. Thus, it is not possible to find a correspondence for these pixels in frame $k+1$. The dotted region in frame $k+1$ indicates the background that is uncovered by the motion of the object. There is no correspondence for these pixels in frame $k$. Note that the roles of covered and uncovered regions are interchanged according to direction of motion estimation. Uncovered regions in backward-correspondence estimation become covered regions in forward estimation, and vice versa.

**Aperture Problem**

The aperture problem is a restatement of the fact that the solution to the 2D motion-estimation problem is not unique. There are many pixels or possibly blocks that are similar to the current pixel or block in the reference picture. Technically speaking, the number of equations (OFE or DFD) is equal to the number of pixels, but the MV has two components for each pixel, and the number of unknowns is twice that of equations. Hence, we can only determine the normal flow. This problem may be overcome if we assume all pixels within a block (at least two pixels) have a common MV. Given a block (aperture) to estimate an MV, there are three possible cases:

- Case 1: There are two linearly independent intensity gradient vectors within the aperture. Both $v_1$ and $v_2$ can be estimated uniquely.
- Case 2: There is only one intensity-gradient direction within the aperture. Only a normal flow (motion) vector can be estimated for the block uniquely.
- Case 3: There is no intensity gradient within the aperture. The motion does not result in observable temporal intensity variation, hence there are infinitely many solutions.

The aperture problem is illustrated in Figure 4.12. Suppose we have a corner of an object moving in the $x_2$ direction (upward). If we estimate the motion based on a local window, indicated by Aperture 1, then it is not possible to determine whether the image moves upward or perpendicular to the edge. Recall that we have shown that the OFE only determines the component of the motion in the direction perpendicular to the edge, called the normal flow.

If we observe Aperture 2, then it is possible to estimate the correct MV, since the image has a gradient in two perpendicular directions. Thus, it is possible to estimate the MV uniquely based on a block of pixels that contain sufficient gray-level variation (gradient) [Hil 84]. Implicit in this discussion is the model that all pixels in the block translate by the same MV.

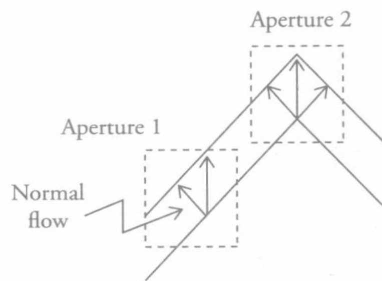



**Figure 4.12** Aperture problem.

The aperture and occlusion problems can be alleviated by employing motion models. Motion can be represented by global models, block models, or dense models. Global and block models can be classified as parametric models, whereas dense models are generally non-parametric.

## 4.3.5 Hierarchical Motion Estimation

The basic idea is to perform motion estimation successively at different levels of the resolution hierarchy, using a coarse-to-fine strategy based on a multi-resolution representation of each frame such as the Gaussian pyramid constructed by repeated blurring and down-sampling (see Section 3.2.3). The Gaussian pyramid representations of the current and reference frames are depicted in Figure 4.13. Optical-flow (displacement) vectors are first computed on the top level (coarsest level with the least number of pixels) and then up-sampled and used to initialize the estimate at the next level. The lower resolution levels serve to determine rough estimates of the displacement that are successively refined at higher resolution levels.

Hierarchical motion estimation has multiple benefits:

1. It is effective in dealing with large motion vectors. At the coarsest level, motion vectors are smaller, helping with the linearization of the OFE.
2. It helps to alleviate the aperture problem, since equal size blocks cover a larger image area at the upper levels of the pyramid, hence reducing the chance of singular image blocks.
3. It helps to reduce the computational complexity, especially in search-based methods. Computation at the higher levels in the pyramid involves fewer pixels, hence is faster. The initialization at each level from the previous level means a smaller range at higher resolution levels and/or fewer iterations are required at each level. Hence, hierarchical methods are faster than single-level methods.

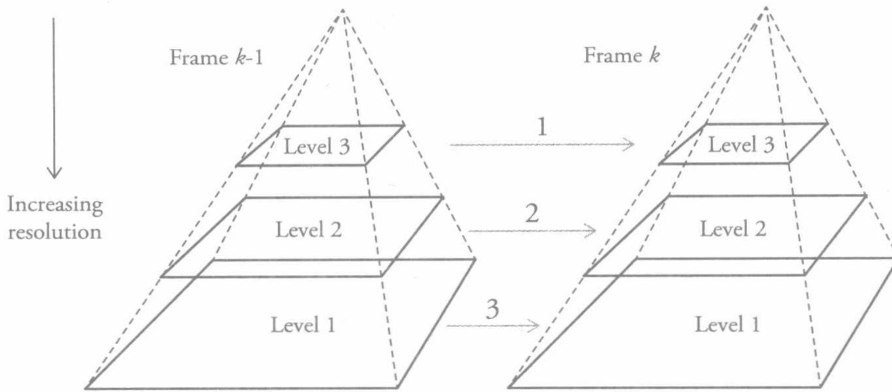



**Figure 4.13** Pyramid representation of each frame.

While hierarchical implementations have many benefits as discussed above, they may fail to follow small objects with fast motion. Hierarchical implementation of differential and matching based motion-estimation methods are discussed in Section 4.4 and Section 4.5, respectively.

## 4.3.6 Performance Measures for Motion Estimation

We can classify error measures for motion estimation as i) those that require ground-truth (GT) motion vectors, such as the norm of the error (NE), also called the end-point error [Bak 11], and the angular error (AE); and ii) those that don't, such as the motion-compensation error (MCE).

Given an estimated motion vector $(d_{i1}, d_{i2})$ at a pixel $\mathbf{x}_i$ and the corresponding GT motion vector $(d_1^{GT}, d_2^{GT})$, the norm of the error can be computed by

$$\| e_i \| = \sqrt{\left(d_{i1} - d_{i1}^{GT}\right)^2 + \left(d_{i2} - d_{i2}^{GT}\right)^2} \tag{4.29a}$$

The angular error between an estimated motion vector $(d_{i1}, d_{i2})$ at a pixel $\mathbf{x}_i$ and the corresponding GT motion vector $(d_1^{GT}, d_2^{GT})$ is the angle $\theta_i$ in between a 3D vector $\mathbf{d}_i = (d_1, d_2, 1.0)$ and $\mathbf{d}_i^{GT} = (d_1^{GT}, d_2^{GT}, 1.0)$, which can be computed by using the dot (inner) product rule

$$\cos\theta_i = \frac{< \mathbf{d}_i, \mathbf{d}_i^{GT} >}{\|\mathbf{d}_i\| \, \|\mathbf{d}_i^{GT}\|} \tag{4.29b}$$

This measure was first proposed by Fleet and Jepson [Fle 90] and has been used in the early comparative study of motion-estimation methods [Bar 94]. Note that the AE penalizes errors in large flows less than errors in small flows.

Motion-compensation error, which requires no ground truth MVs, can be computed as the mean-square error between the actual current image $s_k(x_1, x_2)$ and its predicted version from the reference image $s_{k-1}(x_1, x_2)$ using the estimated motion vectors $(d_1, d_2)$ given by

$$MSE = \sum_{x_1} \sum_{x_2} [s_k(x_1, x_2) - s_{k-1}(x_1 - d_1, x_2 - d_2)]^2 \tag{4.29c}$$

The statistics of pixel-based measures NE $\|e_i\|$ and AE $\theta_i$, including averages, standard deviations, and the percentage of pixels that have an error measure above value X, as well the motion-compensation error have been published to compare recent motion-estimation methods using the Middlebury stereo dataset [Bak 11].

## 4.4 Differential Methods

Motion-estimation methods that utilize the spatial and temporal partial derivatives of images or the OFE are called differential or direct methods. Recall that the estimation of the image gradient has been discussed in Section 3.3. The OFE alone is not sufficient to determine MVs at each pixel since it specifies one equation in two unknowns per pixel (Section 4.3.2). In order to regularize this underdetermined estimation problem, we either employ parametric or non-parametric motion models, which are generally known as the Lukas–Kanade method (Section 4.4.1) and the Horn–Schunk method (Section 4.4.2), respectively.

### 4.4.1 Lukas–Kanade Method

The Lukas–Kanade method [Luc 81] is one of the most popular 2D-motion estimation methods in digital-video processing. Its many extensions include hierarchical model-based motion estimation [Ber 92] and the forward-additive (original), forward-compositional, inverse-additive, and inverse-compositional formulations, which have been shown to be equivalent [Bak 04]. Among these, the inverse compositional algorithm is computationally the most efficient and can be used for direct estimation of most parametric models including homography.

Here, we present the original formulation, where incremental additive parameters are estimated, and we minimize the error in warping the current frame toward the previous frame, given by

$$\sum_{\mathbf{x} \in B} \left[ s_k(\mathbf{x}') - s_{k-1}(\mathbf{x}) \right]^2 \tag{4.30a}$$

where $\boldsymbol{B}$ denotes an $N \times N$ block of pixels with sufficient gray-level variation and $\mathbf{x}' = \boldsymbol{T}(\mathbf{x}; \mathbf{p}) = [T_1(\mathbf{x}; \mathbf{p})\ T_2(\mathbf{x}; \mathbf{p})]^T$ denotes a parametric-motion model as a function of the model parameter vector $\mathbf{p}$. This is a nonlinear optimization problem, which can be approximated by a quadratic cost function by replacing $s_k(\mathbf{x}')$ by its Taylor series expansion given by Eqn. (4.27) assuming small motion. In order to ensure the motion is small, we assume a current estimate of the parameter vector $\mathbf{p}$ is available and consider estimation of differential motion due to incremental parameter update $\Delta\mathbf{p}$; i.e., we expand $s_k(\mathbf{x}')$, where $\mathbf{x}' = \boldsymbol{T}(\mathbf{x}; \mathbf{p} + \Delta\mathbf{p})$, into a Taylor series about the point $\boldsymbol{T}(\mathbf{x}; \mathbf{p})$. The resulting cost function that is quadratic in $\Delta\mathbf{p}$ is given by

$$\sum_{\mathbf{x}} \left[ s_k(T(\mathbf{x};\mathbf{p})) + \nabla s_k(T(\mathbf{x};\mathbf{p})) \frac{\partial T}{\partial p}\ \Delta\mathbf{p} - s_{k-1}(\mathbf{x}) \right]^2 \tag{4.30b}$$

where the term $\frac{\partial T}{\partial p}$ is the Jacobian of the parametric model. Minimization of (4.32b) with respect to $\Delta\mathbf{p}$ is a least-squares estimation problem, which has a closed form solution. Computing the partial derivative of (4.30b) with respect to $\Delta\mathbf{p}$ and setting equal to zero,

$$2\sum_{\mathbf{x}} \left[ \nabla s_k(T(\mathbf{x};\mathbf{p})) \frac{\partial T}{\partial p} \right]^T \left[ s_k(T(\mathbf{x};\mathbf{p})) + \nabla s_k(T(\mathbf{x};\mathbf{p})) \frac{\partial T}{\partial p} \Delta\mathbf{p} - s_{k-1}(\mathbf{x}) \right] = 0$$

Solving for $\Delta\mathbf{p}$, we obtain

$$\Delta\mathbf{p} = \mathbf{H}^{-1} \sum_{\mathbf{x}} \left[ \nabla s_k(T(\mathbf{x};\mathbf{p})) \frac{\partial T}{\partial p} \right]^T \left[ s_{k-1}(\mathbf{x}) - s_k(T(\mathbf{x};\mathbf{p})) \right] \tag{4.31a}$$

where

$$\mathbf{H} = \sum_{\mathbf{x}} \left[ \nabla s_k(T(\mathbf{x};\mathbf{p})) \frac{\partial T}{\partial p} \right]^T \left[ \nabla s_k(T(\mathbf{x};\mathbf{p})) \frac{\partial T}{\partial p} \right] \tag{4.31b}$$

is called the Hessian matrix. These expressions form the basis of the following hierarchical iterative motion-estimation/refinement algorithm [Ber 92, Ira 93].

### Hierarchical Iterative Refinement

The hierarchical iterative-refinement method employs Gaussian pyramids for both frames combined with iterative MV updating within each resolution level to keep the incremental MV updates small, typically under one pixel. At each resolution level, the MV is initialized by that of the previous level multiplied by 2. At any

iteration at a level, the incremental MV is estimated using a reference image that is motion-compensated with the MV from the previous iteration so the incremental MV gets smaller. The iterative-refinement procedure can be summarized as follows:

1. Estimate spatial/temporal image partials on frame $s_k(\mathbf{x})$ at the lowest resolution. Note that partials are estimated only once at each resolution level. Set the initial parameter vector $\mathbf{p} = \mathbf{0}$.
2. Compensate (warp) the current block of frame $s_k(\mathbf{x})$ toward $s_{k-1}(\mathbf{x})$ using the current estimate $\mathbf{p}$ to obtain $s_k(\boldsymbol{T}(\mathbf{x};\mathbf{p}))$ by sub-pixel motion-compensation. Estimate $\Delta\mathbf{p}$ using (4.31).
3. Update $\mathbf{p} = \mathbf{p} + \Delta\mathbf{p}$. Repeat steps 2 and 3 a few times.
4. Proceed to the next resolution level until we reach the highest resolution level in the pyramid. At each resolution level, scale the most recent parameter vector $\mathbf{p}$, estimate spatial/temporal image partials on frame $s_k(\mathbf{x})$, and go to step 2.
5. At the highest resolution level, repeat steps 2 and 3 until the residual parameter update $\Delta\mathbf{p}$ converges to zero.

### Special Case: Block-Translation Motion Model

Here, we work out the scalar equations for the block-translation model, where the parameter vector $\mathbf{p} = (d_1, d_2)$ consists of two displacement values. In this case, the Jacobian $\frac{\partial T}{\partial p}$ is the identity matrix and the above derivation is equivalent to minimizing the error in the OFE. Let's define the error in the optical flow equation at pixel $\mathbf{x}$ as a function of the incremental MV $\Delta\mathbf{p} = (\Delta d_1, \Delta d_2)$ by

$$e_o(\mathbf{x}, \Delta\mathbf{p}) = \frac{\partial s_c(\mathbf{x},t)}{\partial x_1}\Delta d_1 + \frac{\partial s_c(\mathbf{x},t)}{\partial x_2}\Delta d_2 + \frac{\partial s_c(\mathbf{x},t)}{\partial t}\Delta t$$

Note that $e_o(\mathbf{x}, \Delta\mathbf{p})$ is in general not exactly zero at all pixels $\mathbf{x}$ within a block $\boldsymbol{B}$, because i) there may be some errors in estimating the partial derivatives from discrete image samples, ii) there may be multiple motions within a block $\boldsymbol{B}$, and iii) there may be intensity variations from frame to frame. The total square error over a block of pixels $\boldsymbol{B}$ is given by sum of squares of $e_o(\mathbf{x}, \Delta\mathbf{p})$, which can be expressed as

$$E_o(\Delta\mathbf{p}) = \sum_{\mathbf{x}\in\boldsymbol{B}}\left[\frac{\partial s_c(\mathbf{x},t)}{\partial x_1}\Delta d_1 + \frac{\partial s_c(\mathbf{x},t)}{\partial x_2}\Delta d_2 + \frac{\partial s_c(\mathbf{x},t)}{\partial t}\Delta t\right]^2 \qquad (4.32)$$

We can minimize the total square error by computing partials of the error $E_o(\Delta\mathbf{p})$ with respect to unknowns $\Delta d_1$ and $\Delta d_2$, respectively, and setting them equal to zero, which yields two equations, for $i = 1,2$,

$$\frac{\partial E_o(\Delta\mathbf{p})}{\partial \Delta d_i} = 2\sum_{\mathbf{x}\in B}\left[\frac{\partial s_c(\mathbf{x},t)}{\partial x_1}\Delta d_1 + \frac{\partial s_c(\mathbf{x},t)}{\partial x_2}\Delta d_2 + \frac{\partial s_c(\mathbf{x},t)}{\partial t}\Delta t\right]\frac{\partial s_c(\mathbf{x},t)}{\partial x_i} = 0$$

Solving these two equations simultaneously, we have

$$\sum_{\mathbf{x}\in B}\left[\frac{\partial s_c(\mathbf{x},t)}{\partial x_1}\right]^2 \Delta d_1 + \sum_{\mathbf{x}\in B}\left[\frac{\partial s_c(\mathbf{x},t)}{\partial x_1}\right]\left[\frac{\partial s_c(\mathbf{x},t)}{\partial x_2}\right]\Delta d_2 = -\sum_{\mathbf{x}\in B}\left[\frac{\partial s_c(\mathbf{x},t)}{\partial t}\right]\left[\frac{\partial s_c(\mathbf{x},t)}{\partial x_1}\right]\Delta t$$

$$\sum_{\mathbf{x}\in B}\left[\frac{\partial s_c(\mathbf{x},t)}{\partial x_1}\right]\left[\frac{\partial s_c(\mathbf{x},t)}{\partial x_2}\right]\Delta d_1 + \sum_{\mathbf{x}\in B}\left[\frac{\partial s_c(\mathbf{x},t)}{\partial x_2}\right]^2 \Delta d_2 = -\sum_{\mathbf{x}\in B}\left[\frac{\partial s_c(\mathbf{x},t)}{\partial t}\right]\left[\frac{\partial s_c(\mathbf{x},t)}{\partial x_2}\right]\Delta t$$

which can be rewritten in vector-matrix form (by normalizing $\Delta t = 1$) as

$$\mathbf{H}\,\Delta\mathbf{p} = \mathbf{b} \tag{4.33a}$$

where

$$\mathbf{H} = \begin{bmatrix} \sum_{\mathbf{x}\in B}\left[\frac{\partial s_c(\mathbf{x},t)}{\partial x_1}\right]^2 & \sum_{\mathbf{x}\in B}\left[\frac{\partial s_c(\mathbf{x},t)}{\partial x_1}\right]\left[\frac{\partial s_c(\mathbf{x},t)}{\partial x_2}\right] \\ \sum_{\mathbf{x}\in B}\left[\frac{\partial s_c(\mathbf{x},t)}{\partial x_1}\right]\left[\frac{\partial s_c(\mathbf{x},t)}{\partial x_2}\right] & \sum_{\mathbf{x}\in B}\left[\frac{\partial s_c(\mathbf{x},t)}{\partial x_2}\right]^2 \end{bmatrix} \text{ and}$$

$$\mathbf{b} = \begin{bmatrix} -\sum_{\mathbf{x}\in B}\left[\frac{\partial s_c(\mathbf{x},t)}{\partial t}\right]\left[\frac{\partial s_c(\mathbf{x},t)}{\partial x_1}\right] \\ -\sum_{\mathbf{x}\in B}\left[\frac{\partial s_c(\mathbf{x},t)}{\partial t}\right]\left[\frac{\partial s_c(\mathbf{x},t)}{\partial x_2}\right] \end{bmatrix}$$

Then, the estimate of the incremental parameter vector can be computed from

$$\Delta \mathbf{p} = \mathbf{H}^{-1}\mathbf{b} \tag{4.33b}$$

The matrix $\mathbf{H}$ needs to be invertible, i.e., rank 2, for a unique solution, which is satisfied if there is sufficient intensity variation within the block $\boldsymbol{B}$ (see Case 1 under the aperture problem in Section 4.3.4). The solution (4.33) is the least-squares solution of $N \times N$ optical flow equations (4.23), one for each pixel in the block $\boldsymbol{B}$, with the same two unknowns.

The derivation of the Lukas–Kanade solution for the affine model (4.17) or other parametric models that are linear in the unknown parameters, such as (4.18), (4.19), and (4.20), is straightforward and follows the same steps.

### *Estimation of Partials*

Assuming we are estimating motion vectors from frame $k$ to frame $k-1$, the spatial partials are estimated at frame $k$, $s_k(\mathbf{x})$ using the techniques discussed in Section 3.3. The temporal partial can be approximated by the frame difference or estimated by (4.37). Clearly, the accuracy of the motion estimates depends on the accuracy of the estimated spatial and temporal partial derivatives.

### *Spatial Weighting*

It is possible to increase the influence of some pixels in block $\boldsymbol{B}$ by appropriate weighting. A 2D Gaussian or triangular weighting may put higher emphasis on pixels toward the center of a block $\boldsymbol{B}$.

### *Composition of Warps and Computational Complexity*

The compositional approach, proposed in [Shu 00], iteratively solves for an incremental warp $\Delta\mathbf{T}(\mathbf{x};\mathbf{p})$ rather than an additive update to the parameters $\Delta\mathbf{p}$ as in the original Lucas–Kanade formulation. Then, at each iteration, we compute composition of incremental warps, which is equivalent to a bilinear combination of additive update parameters [Bak 04]. The compositional formulation results in a more computationally efficient solution since the Jacobian can be pre-computed and re-used at each iteration.

### *Direct Methods for Homography Estimation*

Direct nonlinear optimization of (4.30) for homography estimation is likely to get stuck at a local minimum without sufficiently close initial estimates [Sze 96]. The

inverse compositional formulation has been shown to provide an efficient solution (see Appendix in [Bak 04]). Linear methods for homography estimation given matched pixel correspondence pairs are addressed in Section 4.5.5.

### 4.4.2 Horn–Schunk Motion Estimation

Horn and Schunck [Hor 81] employ a non-parametric motion model to regularize the ill-posed motion-estimation problem. They seek a motion field that satisfies the OFE with the minimum pixel-to-pixel variation among the flow vectors in order to impose a global smoothness constraint on the velocity field. Hence, representing the spatial and temporal coordinates by continuous variables, motion estimation is posed as a variational optimization problem to minimize a global energy function of the form

$$\hat{\mathbf{v}}(\mathbf{x}) = \arg\min_{\mathbf{v}(\mathbf{x})} \left\{ \int_B \left( E_{of}^2(\mathbf{v}(\mathbf{x})) + \alpha^2 E_s^2(\mathbf{v}(\mathbf{x})) \right) d\mathbf{x} \right\} \tag{4.34a}$$

where $\mathbf{B}$ denotes continuous image support, and

$$E_{of}(\mathbf{v}(\mathbf{x})) = \langle \nabla s_c(\mathbf{x},t), \mathbf{v}(\mathbf{x}) \rangle + \frac{\delta s_c(\mathbf{x},t)}{\delta t} \tag{4.34b}$$

is the error in the optical flow equation, which imposes data consistency. The second term $E_s(\mathbf{v}(\mathbf{x}))$ is a smoothness prior, where pixel-to-pixel variation of the velocity vectors can be quantified by the sum of magnitude squares of the spatial gradients of the components of velocity vector, given by

$$\begin{aligned} E_s^2(\mathbf{v}(\mathbf{x})) &= \| \nabla v_1(\mathbf{x}) \|^2 + \| \nabla v_2(\mathbf{x}) \|^2 \\ &= \left( \frac{\delta v_1(\mathbf{x})}{\delta x_1} \right)^2 + \left( \frac{\delta v_1(\mathbf{x})}{\delta x_2} \right)^2 + \left( \frac{\delta v_2(\mathbf{x})}{\delta x_1} \right)^2 + \left( \frac{\delta v_2(\mathbf{x})}{\delta x_2} \right)^2 \end{aligned} \tag{4.34c}$$

It can easily be verified that the smoother the velocity field, the smaller $E_s^2(\mathbf{v}(\mathbf{x}))$. The parameter $\alpha^2$, usually selected heuristically, controls the strength of the smoothness constraint. Larger values $\alpha^2$ increase the influence of the constraint.

Minimization of the functional (4.34) is treated as a calculus of variations problem leading to the Euler-Lagrange equations given by

$$\frac{\delta E(\mathbf{v}(\mathbf{x}))}{\delta v_1} - \frac{\delta}{\delta x_1} \frac{\delta E(\mathbf{v}(\mathbf{x}))}{\delta v_{1,x_1}} - \frac{\delta}{\delta x_2} \frac{\delta E(\mathbf{v}(\mathbf{x}))}{\delta v_{1,x_2}} = 0 \tag{4.35a}$$

$$\frac{\delta E(\mathbf{v}(\mathbf{x}))}{\delta v_2} - \frac{\delta}{\delta x_1}\frac{\delta E(\mathbf{v}(\mathbf{x}))}{\delta v_{2,x_1}} - \frac{\delta}{\delta x_2}\frac{\delta E(\mathbf{v}(\mathbf{x}))}{\delta v_{2,x_2}} = 0 \tag{4.35b}$$

where $v_{1,x_1} = \delta v_1 / \delta x_1$, $v_{1,x_2} = \delta v_1 / \delta x_2$, $v_{2,x_1} = \delta v_2 / \delta x_1$, and $v_{2,x_2} = \delta v_2 / \delta x_2$. These equations, which are linear in the unknowns $v_1$ and $v_2$, can be solved by the Gauss-Seidel iterations,

$$\hat{v}_1^{(n+1)}(\mathbf{x}) = \hat{v}_1^{(n)}(\mathbf{x}) - \frac{\delta s_c(\mathbf{x},t)}{\delta x_1}\left[\frac{\frac{\delta s_c(\mathbf{x},t)}{\delta x_1}\hat{v}_1^{(n)}(\mathbf{x}) + \frac{\delta s_c(\mathbf{x},t)}{\delta x_2}\hat{v}_2^{(n)}(\mathbf{x}) + \frac{\delta s_c(\mathbf{x},t)}{\delta t}}{\alpha^2 + \left(\frac{\delta s_c(\mathbf{x},t)}{\delta x_1}\right)^2 + \left(\frac{\delta s_c(\mathbf{x},t)}{\delta x_2}\right)^2}\right] \tag{4.36a}$$

$$\hat{v}_2^{(n+1)}(\mathbf{x}) = \hat{v}_2^{(n)}(\mathbf{x}) - \frac{\delta s_c(\mathbf{x},t)}{\delta x_2}\left[\frac{\frac{\delta s_c(\mathbf{x},t)}{\delta x_1}\hat{v}_1^{(n)}(\mathbf{x}) + \frac{\delta s_c(\mathbf{x},t)}{\delta x_2}\hat{v}_2^{(n)}(\mathbf{x}) + \frac{\delta s_c(\mathbf{x},t)}{\delta t}}{\alpha^2 + \left(\frac{\delta s_c(\mathbf{x},t)}{\delta x_1}\right)^2 + \left(\frac{\delta s_c(\mathbf{x},t)}{\delta x_2}\right)^2}\right] \tag{4.36b}$$

where spatial partials are evaluated at $(\mathbf{x}, t)$. The complete derivation can be found in [Hor 81]. The initial estimates of the velocities $v_1^{(0)}(\mathbf{x},t)$ and $v_2^{(0)}(\mathbf{x},t)$ are usually set to zero, and all spatial and temporal partials are estimated from the observed images. Horn and Schunck [Hor 81] proposed to estimate both spatial and temporal partials by averaging four finite differences, where the temporal partials are estimated from two frames by

$$\begin{aligned}\frac{\delta s_c(x_1,x_2,t)}{\delta t} \approx \frac{1}{4}\{&s[n_1,n_2,k+1] - s[n_1,n_2,k] + s[n_1+1,n_2,k+1] \\ &- s[n_1+1,n_2,k] + s[n_1,n_2+1,k+1] - s[n_1,n_2+1,k] \\ &+ s[n_1+1,n_2+1,k+1] - s[n_1+1,n_2+1,k]\}\end{aligned} \tag{4.37}$$

Other methods to estimate spatial-image partials, e.g., using derivatives of Gaussian filters, have been discussed in Section 3.3.

The Horn–Schunck method imposes optical-flow and smoothness constraints globally over the entire image, or over a selected window. This has some undesired effects:

1. A global smoothness constraint blurs "motion edges." For example, if an object moves against a stationary background, there is a sudden change in the motion

field at the boundary of the object. Motion edges can be preserved by imposing the smoothness constraint along object boundaries but not perpendicular to motion boundaries. This is the basic concept of the so-called directional or oriented smoothness constraints, discussed next.

2. The optical-flow constraint has also been enforced at the occlusion regions, where it is indeed not valid. The OFE must be enforced selectively by varying $\alpha$ adaptively to control the relative strengths of the optical-flow and smoothness constraints. For example, at occlusion regions, such as the dotted regions shown in Figure 4.11, the optical-flow constraint should be turned off, while the smoothness constraint must remain fully on.

**Adaptive Smoothness Constraints**

Several researchers proposed to impose adaptive smoothness constraints. Hildreth [Hil 84] minimized the criterion function of Horn and Schunck given by (4.34) along object contours. Nagel and Enkelman [Nag 86, Enk 88] introduced a directional smoothness constraint, which suppresses the smoothness constraint in the direction of the spatial-image gradient. Fogel [Fog 91] used the directional smoothness constraint with adaptive weighting in a hierarchical formulation. Note that adaptive weighting methods require strategies to detect moving object (occlusion) boundaries. Snyder [Sny 91] proposed a general formulation of the smoothness constraint that includes some of the above as special cases. The directional smoothness constraint can be expressed as

$$E_{ds}^2(\mathbf{v}(\mathbf{x})) = (\nabla v_1)^{\mathrm{T}}\mathbf{W}\,(\nabla v_1) + (\nabla v_2)^{\mathrm{T}}\mathbf{W}\,(\nabla v_2) \tag{4.38}$$

where $\mathbf{W}$ is a weight matrix to penalize variations in the motion field depending on the spatial changes in gray-level content of the video. Various alternatives for the weight matrix $\mathbf{W}$ exist [Nag 86, Nag 87, Enk 88]. For example, $\mathbf{W}$ can be chosen as

$$\mathrm{W} = \frac{\mathbf{F} + \delta\mathbf{I}}{\mathrm{trace}(\mathbf{F} + \delta\mathbf{I})}$$

where $\mathbf{I}$ is the identity matrix to ensure a non-zero weight matrix at spatially uniform regions, $\delta$ and $b^2$ are global scalar constants, and

$$\mathbf{F} = \begin{bmatrix} \left(\frac{\delta s_c}{\delta x_1}\right)^2 + b^2\left\{\left(\frac{\delta^2 s_c}{\delta x_1^{\,2}}\right)^2 + \left(\frac{\delta^2 s_c}{\delta x_1 \delta x_2}\right)^2\right\} & \frac{\delta s_c}{\delta x_1}\frac{\delta s_c}{\delta x_2} + b^2\left\{\frac{\delta^2 s_c}{\delta x_1 \delta x_2}\left(\frac{\delta^2 s_c}{\delta x_1^{\,2}}\right)^2 + \left(\frac{\delta^2 s_c}{\delta x_1 \delta x_2}\right)^2\right\} \\ \frac{\delta s_c}{\delta x_1}\frac{\delta s_c}{\delta x_2} + b^2\left\{\frac{\delta^2 s_c}{\delta x_1 \delta x_2}\left(\frac{\delta^2 s_c}{\delta x_1^{\,2}}\right)^2 + \left(\frac{\delta^2 s_c}{\delta x_1 \delta x_2}\right)^2\right\} & \left(\frac{\delta s_c}{\delta x_2}\right)^2 + b^2\left\{\left(\frac{\delta^2 s_c}{\delta x_2^{\,2}}\right)^2 + \left(\frac{\delta^2 s_c}{\delta x_1 \delta x_2}\right)^2\right\} \end{bmatrix}^{-1}$$

Observe that the method of Horn and Schunck (4.34) is a special case of this formulation with $\delta = 1$ and $\mathbf{F} = \mathbf{0}$. A Gauss–Seidel iteration to minimize the problem formulation using (4.44) has been described in [Enk 88], where the update term at each iteration is computed by means of a linear algorithm. The performance of the directional-smoothness method depends on how accurately the required second and mixed partials of image intensity can be estimated. A hierarchical implementation of adaptive smoothness constraints can be found in [Fog 91].

#### Median Filtering as an Energy Function

Sun *et al.* [Sun 10] observed that median filtering of the intermediate flow results, once after every iteration, e.g., a $5 \times 5$ median filter, results in significantly better results, although this leads to higher energy solutions. Hence, they propose a new objective function that formalizes the heuristic median filtering. This objective function includes a non-local term that robustly integrates flow estimates over large spatial neighborhoods. A hierarchical estimation procedure has been proposed, where they alternate between minimizing a classical Horn–Schunk type of energy function and a new median-filtering energy function 10 times at every level of the pyramid.

## 4.5 Matching Methods

We can classify matching methods as i) block-matching, which assigns a forward and/or backward-motion vector to blocks of pixels, where blocks may be overlapping or non-overlapping, and ii) sparse feature-matching methods. Block-matching

motion estimation, which can be performed using fixed or variable size blocks, is commonly employed in video-compression standards such as ISO/IEC MPEG-2, which uses fixed-size blocks, and advanced video coding (AVC) and high-efficiency video coding (HEVC) (ITU-T H.264/265), which allow variable size blocks. Variable block-size motion-estimation offers a tradeoff in video coding such that larger fixed size blocks reduces the number of bits needed to encode MVs at the expense of an increase in the number of bits to encode the prediction residual, while using smaller variable size blocks can result in a reduction in the number of bits needed to encode prediction residual at the expense of an increase in the number of bits to encode MVs. Hierarchical block-matching is often preferred since it increases estimation accuracy and helps reduce search complexity. Sparse feature matching aims to match a predetermined set of feature points between pairs of frames, which can be later used in feature-point tracking or to determine parameters of a motion model from some number of feature correspondences.

### 4.5.1 Basic Block-Matching

The basic block-matching procedure takes a fixed size block from the present frame and searches for the location of the best-matching block of the same size in a (past and/or future) reference frame based on some distance criterion (see Figure 4.14). Block-matching algorithms differ in the choice of the block size, matching (distance) criteria, and search strategy employed.

The matching error can be quantified according to several criteria including minimum mean-square error (MSE), minimum mean absolute difference (MAD), maximum cross-correlation, maximum matching pel count (MPC), and so on. The MSE criterion is defined by

$$MSE(d_1, d_2) = \frac{1}{N_1 N_2} \sum\sum_{(n_1, n_2) \in B} \left| s[n_1, n_2, k] - s[n_1 + d_1, n_2 + d_2, k-1] \right|^2 \tag{4.39}$$

where $\boldsymbol{B}$ is an $N_1 \times N_2$ block, and $(d_1, d_2)$ denotes a candidate MV. The MAD criterion, defined by

$$MAD(d_1, d_2) = \frac{1}{N_1 N_2} \sum\sum_{(n_1, n_2) \in B} \left| s[n_1, n_2, k] - s[n_1 + d_1, n_2 + d_2, k-1] \right| \tag{4.40}$$

is the most popular choice for very large scale integration (VLSI) implementations.

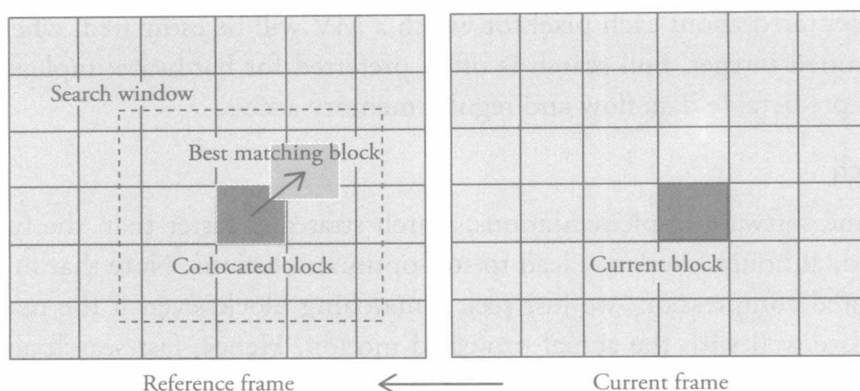



**Figure 4.14** Block-matching.

The estimated MV is the value of $(d_1, d_2)$ that minimizes (4.39) or (4.40)

$$[\hat{d}_1 \ \hat{d}_2]^T = \arg \ \min_{(d_1,d_2)} MSE(d_1, d_2)$$

or

$$[\hat{d}_1 \ \hat{d}_2]^T = \arg \ \min_{(d_1,d_2)} MAD(d_1, d_2)$$

The performance of the MAD criterion may deteriorate as the search area becomes larger due to presence of several local minima.

For video coding, every frame is partitioned into fixed or variable size blocks and one (for uni-directional) or two (for bi-directional) MV is computed for each block. The block sizes may vary between $8 \times 8$ and $64 \times 64$ (in the HEVC standard). For other applications, a common approach to computing a dense-motion field using block-matching is to estimate motion vectors on a sparse grid of pixels, e.g., once every four pixels and four lines with partially overlapping blocks of size $N_1 = N_2 = 16$, and then interpolating the remaining vectors to obtain a dense motion field.

### Full Search

Finding the best-matching block requires computation of the matching criterion for all candidate motion vectors $(d_1, d_2)$ at each pixel $(n_1, n_2)$. This procedure, called "full search," is time-consuming. In order to reduce the computational load, we can limit candidate motion vectors to within a $(2M+1) \times (2M+1)$ "search window" (depicted in Figure 4.14) such that

$$-M \leq d_1 \leq M \text{ and } -M \leq d_2 \leq M$$

which is centered about each pixel for which a MV will be estimated, where $M$ is a predetermined integer. Full search is often preferred for hardware implementation due to its predictable data flow and regular memory access.

**Fast Search**

In real-time software implementations, search strategies faster than the full search are needed, although they may lead to sub-optimal solutions. Note that in motion-compensated compression, we just seek a matching block, even if the match does not correlate well with the actual projected motion. Hence, fast-search algorithms serve video-compression applications reasonably well. Fast search algorithms can be classified as: i) those that eliminate some candidate MVs based on mathematical lower bounds and give the same result as that of the full search, e.g., the successive elimination method and its improvements; ii) those that evaluate the criterion function only at a subset of candidate motion vectors, e.g., logarithmic search, three-step search, and diamond search; and iii) early-termination methods that terminate the search procedure, for example, by predicting zero motion vector or zero transform coefficients for some blocks [Yan 05].

The successive elimination algorithm (SEA) [Li 95] is based on the triangle inequality

$$\sum\sum_{(n_1,\, n_2)\,\in B} |s[n_1,n_2,k]| - \sum\sum_{(n_1,\, n_2)\,\in B} |s[n_1+d_1,n_2+d_2,k-1]| \leq$$
$$\sum\sum_{(n_1,\, n_2)\,\in B} |s[n_1,n_2,k]-s[n_1+d_1,n_2+d_2,k-1]| = SAD(d_1,d_2)$$

where $SAD(d_1, d_2)$ denotes the sum of absolute differences. Hence, the difference between the sum of intensity values and the sum of displaced intensity values, which can be efficiently computed using the box-filtering technique, establishes a lower bound for the SAD value. In the SEA, the difference on the left-hand side for each candidate MV is compared to the previous minimum SAD value, and the SAD computation is skipped if the lower bound is greater than the previous minimum.

Logarithmic search proposed by Jain and Jain [Jai 81] and three-step search (TSS) proposed by Koga *et al.* [Kog 81] are both multi-step search procedures that define an initial step size and a set of search points centered at the center of the search window and terminate when the step size reduces to one. They both have complexity $O(\log(M/2))$ but the logarithmic search is generally more accurate. The logarithmic search begins with calculating SAD at the center of the search window (zero motion vector) and four points that are $\pm P$ pixels, where typically $P = M/2$, from the center in the horizontal and

vertical directions that are marked with 1 in Figure 4.15. If the minimum SAD occurs at the center, then the step size is halved. Otherwise, the step size remains unchanged, the next search is centered about the pixel with the minimum SAD, and SAD values at new four points $\pm P$ pixels from the new center (marked with 2 in Figure 4.15) are calculated. When the step size becomes 1, the nine neighbors about the current center are searched and the pixel with the minimum SAD defines the final integer MV.

The diamond search (DS), proposed by Zhu and Ma [Zhu 00], employs two diamond-shaped search patterns: a large diamond with nine search points and a small diamond with five points, that are shown in Figure 4.16(a). DS starts on the large diamond centered at the center of the search window and searches for the minimum SAD location. If the minimum SAD does not occur at the center, the next search is centered at the pixel with the minimum SAD value and the search continues with the large diamond pattern until the minimum occurs at the center of the large diamond. When the minimum occurs at the center, the final search is conducted using the small diamond centered about this center pixel, and the location of the minimum SAD on the small diamond defines the final integer MV. The complexity of DS is $O(\log(M/2))$ and it has better accuracy than logarithmic search and TSS.

Recently, more sophisticated fast-motion estimation schemes, such as UMHexagonS [Xu 08], which include: i) initial search-point prediction (rather than starting at the center of the search window); ii) combination of multiple search schemes, such as cross-search, uneven multi-hexagon search and diamond search; and iii) early termination criteria have been proposed.

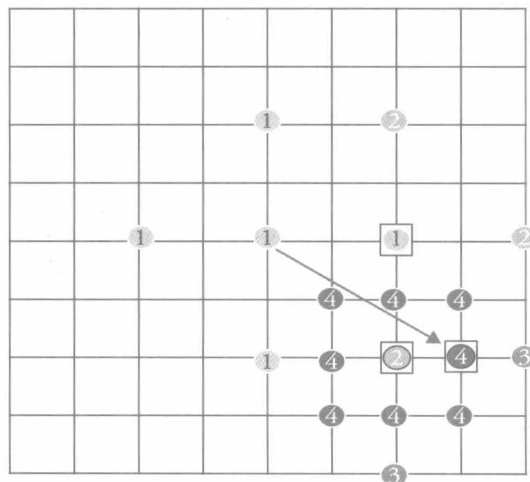



**Figure 4.15** Four-step logarithmic search. The best match position at the end of each step is indicated by a square. The arrow shows the final motion vector.

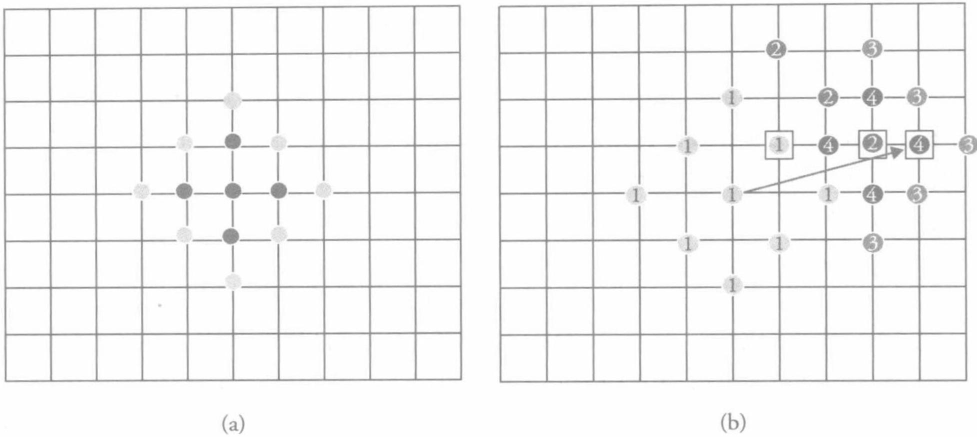



**Figure 4.16** Diamond search: (a) big diamond (light) and small diamond (dark) centered about the center of the search window and (b) a four-step diamond search, where the first three steps use big diamond and the last step uses small diamond to find an integer estimate (arrow). The best match position at the end of each step is indicated by a square.

### Sub-Pixel Search

In most applications, motion vectors are estimated by higher than integer pixel precision, called *sub-pixel* precision. For example, most video-compression schemes employ half-pixel or quarter-pixel precision MVs. The computational complexity of sub-pixel motion estimation is higher due to the interpolation required to compute in-between pixels and a larger number of candidate blocks that need to be evaluated. Hence, sub-pixel search is conducted only in the vicinity of the best integer MV, which is estimated first. We evaluate SAD at the eight half-pixel positions around the best integer MV (depicted in Figure 4.17) to check whether it can be lowered. If needed, we next evaluate SAD at the eight quarter-pixel positions around the best half-pixel MV to find the best quarter pixel MV. Typically, half-pixel sample values are evaluated by using a separable six-tap FIR filter horizontally and vertically, while quarter-pixel sample values are computed by bilinear interpolation between full and computed half-pixel samples. It has been shown that the filters used for interpolation have an impact on the accuracy of the estimated MVs.

## 4.5.2 Variable-Size Block-Matching

In block-motion estimation, it is assumed that all pixels within a block move uniformly that can be described by a single, common motion vector. Blocks containing

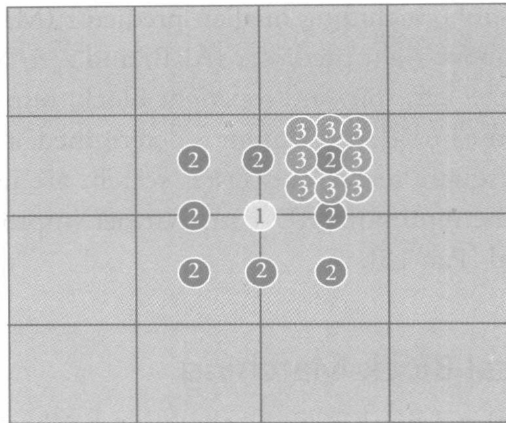



**Figure 4.17** Sub-pixel search locations: (1) best integer MV with the grid indicating pixels; (2) half-pixel search positions; and (3) quarter-pixel search positions about the best half-pixel MV.

motion edges pose a challenge in fixed block-size motion estimation since multiple distinct motion vectors may be needed. A convenient approach to overcome this problem is to subdivide such blocks so that pixels within each subblock, which can be as small as $4 \times 4$, have a single MV resulting in so-called variable size block-matching (VSBM). This subdivision is usually performed using a quad-tree for efficient representation of the block partitioning. In VSBM, smaller blocks are used in image regions with complex motion, while larger blocks can be used where the image is stationary or undergoes uniform motion. VSBM algorithms provide the ability to dynamically adapt the block size to the nature of the motion field and consist of two steps that are coupled: i) selecting the best partitioning of a square block of pixels, and ii) finding the best motion vector for each sub-block.

The main idea in efficient implementation of VSBM is to reuse SAD computations as much as possible, including i) compute SAD for the smallest sub-block size (typically $4 \times 4$) and then compute SAD for larger blocks by summing the appropriate combination of these SAD values, and ii) reuse partial SAD values for one candidate MV for computing SAD for other candidate MVs. Several architectures have been proposed for hardware implementation of VSBM, including FPGA, 1D and 2D systolic arrays, and ASIC implementations.

Among software solutions, the Test Zone Search (TZSearch) [Pur 12] algorithm was adopted in HEVC (the most recent video-compression standard) reference software as a fast ME algorithm for reducing search time with an RD performance comparable to that of full search. The TZSearch includes two steps: determination of initial search point and a search procedure. The initial search point is determined

by using a set of predictors, including median predictor (MP), left predictor (LP), above predictor (AP), above-right predictor (ARP) and zero MV (0,0). The LP, AP, and ARP take MV of the left, top, and top-right block, respectively. MP takes the median of them. After the initial search point is determined, a hybrid search, including multiple diamond/square search and raster search, are used to locate the best matching block with the minimum RD cost. Further improvements to TZSearch have also been proposed [Pan 13].

### 4.5.3 Hierarchical Block-Matching

The basic principle of hierarchical-motion estimation has been discussed in Section 4.3.5. In hierarchical block-matching, we compute Gaussian pyramid representation for both frames and perform search successively at different levels of the hierarchy, starting with the lowest resolution level. At lower resolution levels, rougher MV estimates are determined using relatively larger blocks, where "relative size of the block" is measured as the size of the block normalized by the size of the image at that resolution level. The estimate of MV at a lower resolution level is then passed onto the next higher resolution level as an initial estimate. Higher resolution levels serve to finetune the MV estimate with a relatively smaller window size and a good initial estimate. If more than one MV yields similar SAD at the lower resolution levels, then they can all be refined at higher levels and the best MV is selected among them at the end. The increase in computational complexity is small compared to that in the quality of the results.

Figure 4.18 illustrates hierarchical block-matching with two levels, where the search range $M = 7$ for Level 2 (lower resolution) and $M = 3$ for Level 1 (higher resolution) [Bie 88]. For simplicity, we assume that images at all levels of the pyramid are the same size but successively more blurred as we go to lower resolution levels. In the two-level pyramid, we simply skip over every other pixel in the low-resolution level (when computing matching criterion) to simulate the effect of sub-sampling. The best estimate at the lowest resolution level is indicated by the circled "3." The center of the search area in Level 1 (denoted by "0") corresponds to the best estimate from the second level. The estimates in the low and high levels are $[7,1]^T$ and $[3,1]^T$, respectively, resulting in an overall estimate of $[10,2]^T$. Half-pixel and quarter-pixel search can also be performed about the integer MV by using appropriate interpolation filters.

We note that the SEA algorithm for faster full search has been extended to hierarchical-motion estimation and is called multi-level SEA or MSEA, whose computational efficiency is further improved by eliminating some redundant terms in the test condition [Ahn 03].

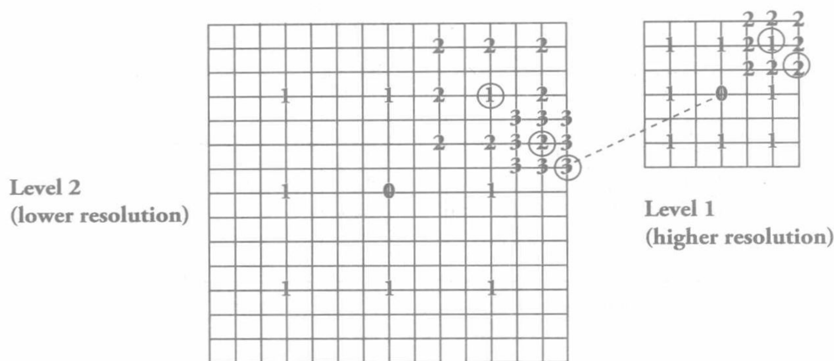



**Figure 4.18** Example of hierarchical block-matching with two levels.

## 4.5.4 Generalized Block-Matching – Local Deformable Motion

Spatial transformations via parametric models, such as (4.17) to (4.20), provide superior image registration and rendering, especially in the presence of local deformations, compared to block translation model (4.16). Such local deformable motion can be modeled by 2D rectangular or triangular mesh models with or without connectivity constraints.

The concept of block-matching can be extended to estimate the parameters of these more sophisticated motion models. This, of course, requires higher computational complexity; we now have to perform search in 6D (affine) or 8D (perspective) parameter space instead of a 2D space. Here, we present two generalized matching schemes: the full search, which does not impose connectivity constraints, and hexagonal matching, which does. Connectivity is not valid at motion/occlusion boundaries. Hence, in practice a combination of both approaches, where connectivity is turned off at occlusion boundaries, should be preferred.

The full-search method, without connectivity constraints, can be summarized as:

1. Segment the current frame into rectangular blocks as illustrated in Figure 4.19.
2. Perturb coordinates of the four corners of the co-located block in the reference frame starting from an initial guess to form a candidate-matching quadrilateral.
3. For each candidate, find the model parameters that map this quadrilateral onto the rectangular block in the current frame using the four corners as feature correspondences.
4. Perform the spatial transformation using the computed parametric model, and calculate the MSE between the given block and the matching quadrilateral.
5. Choose the spatial transformation that yields the smallest MSE or MAD.

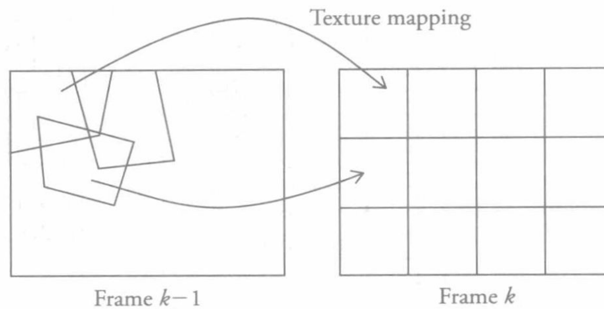



**Figure 4.19** Motion compensation without connectivity constraints on a 2D rectangular mesh.

In order to reduce the computational burden imposed by generalized block-matching, it is only used for those blocks where standard block-matching is not satisfactory. The displaced frame difference resulting from standard block-matching can be used as a decision criterion.

A connectivity preserving motion-estimation method applied to 2D triangular meshes, called hexagonal matching, was proposed by Nakaya and Harashima [Nak 94]. The hexagonal search is based on the observation that there are six lines intersecting at each node in a uniform triangular mesh, and the boundaries of these six triangles define a hexagon as depicted in Figure 4.20. Assuming the motion vector is constrained to stay inside this hexagon, each node (in the reference frame) is perturbed one at a time to find the best matching hexagon between the current and reference frames. The SAD for a hexagon is computed by mapping the texture within each of six triangles affected by a perturbed node by their respective affine parameters. Later, Toklu *et al.* [Tok 96] and Altunbasak *et al.* [Alt 97] have proposed improvements to mesh-based motion estimation.

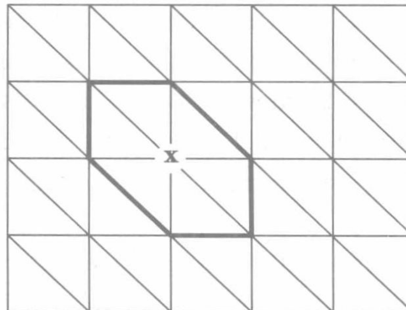



**Figure 4.20** Motion compensation with connectivity constraint on a 2D triangular mesh.

### 4.5.5 Homography Estimation from Feature Correspondences

This section presents methods for homography estimation from a number of pre-determined feature correspondences. Detection of "good" features and precision (sub-pixel) of correspondence estimation play important roles in the accuracy of the parameter estimates. For each pre-determined feature point $i$, we can express the projective transformation (4.14) in the homogeneous coordinates $\mathbf{z}$, where $\mathbf{z}_i = [z_{i1} \quad z_{i2} \quad z_{i3}]^{\mathrm{T}}$ and $x_{i1} = z_{i1}/z_{i3}$, $x_{i2} = z_{i2}/z_{i3}$

$$\mathbf{z}'_i = \begin{bmatrix} z'_{i1} \\ z'_{i2} \\ z'_{i3} \end{bmatrix} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} x_{i1} \\ x_{i2} \\ 1 \end{bmatrix} = \mathbf{H}\,\mathbf{z}_i, \; i = 1, \ldots, N$$

where $z_{i3}$ is set equal to 1. Observe that $\mathbf{H}$ can be multiplied by a non-zero constant without altering the image coordinates $x'_{i1} = z'_{i1} / z'_{i3}$ and $x'_{i2} = z'_{i2} / z'_{i3}$, which is referred to as scale ambiguity. Thus, $\mathbf{H}$ is a homogeneous matrix with only 8 degrees of freedom even though it has 9 entries. Expressing

$$x'_{i1} = \frac{z'_{i1}}{z'_{i3}} = \frac{h_1 x_{i1} + h_2 x_{i2} + h_3}{h_7 x_{i1} + h_8 x_{i2} + h_9} \quad \text{and} \quad x'_{i2} = \frac{z'_{i2}}{z'_{i3}} = \frac{h_4 x_{i1} + h_5 x_{i2} + h_6}{h_7 x_{i1} + h_8 x_{i2} + h_9}$$

and cross-multiplying both equations, we have

$$\begin{aligned} h_7 x'_{i1} x_{i1} + h_8 x'_{i1} x_{i2} + h_9 x'_{i1} - h_1 x_{i1} - h_2 x_{i2} - h_3 &= 0 \\ h_7 x'_{i2} x_{i1} + h_8 x'_{i2} x_{i2} + h_9 x'_{i2} - h_4 x_{i1} - h_5 x_{i2} - h_6 &= 0 \end{aligned}$$

which can be rewritten in vector-matrix form as

$$\mathbf{A}_i \mathbf{h} = \begin{bmatrix} -x_{i1} & -x_{i2} & -1 & 0 & 0 & 0 & x'_{i1} x_{i1} & x'_{i1} x_{i2} & x'_{i1} \\ 0 & 0 & 0 & -x_{i1} & -x_{i2} & -1 & x'_{i2} x_{i1} & x'_{i2} x_{i2} & x'_{i2} \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \end{bmatrix} = \mathbf{0} \quad (4.41a)$$

Hence, we obtain two linearly independent equations for each feature point correspondence, excluding the point at infinity, i.e., assuming $z'_{i3} \neq 0$.

**Direct Linear Transformation (DLT) Method**

Given $N \geq 4$ feature-point correspondences, such that no three points are collinear, we can obtain $2N$ homogeneous linear equations in the form

$$\begin{bmatrix} \mathbf{A}_1 \\ \vdots \\ \mathbf{A}_N \end{bmatrix} \mathbf{h} = \mathbf{A}\,\mathbf{h} = \mathbf{0} \tag{4.41b}$$

where $\mathbf{A}$ is a $2N \times 9$ matrix and $\mathbf{h}$ is a $9 \times 1$ vector consisting of unknown homography parameters.

If $N = 4$ or $N > 4$ but all point correspondences are exact (noise-free), then $\mathbf{A}$ has rank 8. Hence, $\mathbf{A}$ has a 1D null space that provides a solution for $\mathbf{h}$ that can only be determined up to a non-zero scale factor. Recall that $\mathbf{H}$ is defined up to a scale factor anyway. A scale factor may arbitrarily be chosen by setting $\|\mathbf{h}\| = 1$, which avoids the trivial solution $\mathbf{h} = \mathbf{0}$. If $N > 4$ and the correspondences are noisy, then the overdetermined system $\mathbf{A}\,\mathbf{h} = \mathbf{0}$ is inconsistent and does not have a solution. We can then find a least-squares solution for $\mathbf{h}$, which minimizes $\|\mathbf{A}\,\mathbf{h}\|$, subject to $\|\mathbf{h}\| = 1$. In either case, $\mathbf{h}$ is given by the last column of $\mathbf{V}$, where $\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\mathrm{T}}$ is the singular value decomposition (SVD) of $\mathbf{A}$ (see Appendix D).

An alternative derivation of the DLT method that is based on the fact that $\mathbf{z}'_i \times \mathbf{H}\,\mathbf{z}_i = \mathbf{0}$ and keeps all three equations to include the case $z'_{3i} = 0$ (a point at infinity) when dehomogenization leading to Eqn. (4.41a) is not possible can be found in [Har 04].

We note that it is possible to estimate the parameters of the homography (4.14) by setting $h_9 = 1$ as a scale parameter first and solving the resulting set of inhomogeneous equations. However, this approach gives poor results if the actual value of $h_9$ is close to 0, and hence is not recommended.

**Normalization**

The performance of the basic DLT algorithm depends on the origin and scale of the coordinate system for both image frames. Hence, normalization of the pixel coordinates helps to obtain numerically stable solutions [Har 04]. The normalized DLT algorithm works as follows:

1. Compute a similarity transform for the first frame $\tilde{\mathbf{x}}_i = \mathbf{T}\mathbf{x}_i$ such that the origin of the new coordinate system is the centroid of points $\tilde{\mathbf{x}}_i$ and the average distance of points $\tilde{\mathbf{x}}_i$ from the origin is $\sqrt{2}$.
2. Compute a similarity transform $\tilde{\mathbf{x}}_i' = \mathbf{T}'\mathbf{x}_i'$ for the second frame the same way, independent of the first frame.
3. Apply the DLT algorithm using the point correspondences $(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_i')$ to obtain homography $\tilde{\mathbf{H}}$.
4. The desired homography matrix is given by $\mathbf{H} = (\mathbf{T}')^{-1}\tilde{\mathbf{H}}\mathbf{T}$.

The normalized DLT method, also called the normalized 8-point algorithm, provide quite satisfactory results [Har 97].

**Other Cost Functions**

The solution of the overdetermined linear homogeneous equations (4.41b) is an optimization problem. The DLT method minimizes the so-called algebraic error $\|\mathbf{Ah}\|$, subject to $\|\mathbf{h}\| = 1$. Other cost functions that have been considered include the geometric error (uni-directional or bi-directional transfer error), the reprojection error, and the Sampson error [Har 04]. However, minimization of these alternate cost functions result in more complex iterative estimation methods and will not be considered here.

**Other Models**

Parameters of bi-linear or quadratic models can similarly be estimated from at least four point correspondences. An affine model can be estimated from at least three point correspondences. A comprehensive overview of parametric-motion estimation methods can be found in [Sze 06].

## 4.6 Nonlinear Optimization Methods

This section discusses methods which employ nonlinear optimization schemes for 2D-motion estimation. Pel-recursive motion estimation that uses gradient descent minimization is presented in Section 4.6.1. Bayesian motion estimation that uses probabilistic smoothness priors and nonlinear optimization to compute the *maximum a posteriori* (MAP) estimate of motion field is introduced in Section 4.6.2.

### 4.6.1 Pel-Recursive Motion Estimation

Pel-recursive methods are predictor-corrector type estimators computed sequentially at each pixel. Pel-recursive motion estimation is usually preceded by a

change-detection stage, where the frame difference at each pixel is tested against a threshold. Estimation is performed only at those pixels belonging to the changed region.

An early pel-recursive approach is the Netravali-Robbins algorithm [Rob 83], which minimizes the square of the DFD at each pixel, given by

$$E(\mathbf{x},\mathbf{d}) = [DFD(\mathbf{x},\mathbf{d})]^2 \tag{4.42}$$

where $DFD(\cdot)$ denotes the displaced frame difference. Minimization of $E(\mathbf{x},\mathbf{d})$ with respect to $\mathbf{d}$, at pixel $\mathbf{x}$, by the steepest descent method yields the iteration

$$\hat{\mathbf{d}}^{(i+1)}(\mathbf{x}) = \hat{\mathbf{d}}^{(i)}(\mathbf{x}) - \varepsilon DFD(\mathbf{x},\hat{\mathbf{d}}^{(i)})\ \nabla_{\mathbf{x}} s_c(\mathbf{x}-\hat{\mathbf{d}}^{(i)};t-\Delta t) \tag{4.43}$$

where $\nabla_{\mathbf{x}}$ is the gradient with respect to $\mathbf{x}$, and is $\varepsilon$ the step size. The initial estimate $\hat{\mathbf{d}}^{(0)}(\mathbf{x})$ can be taken as the MV at the previous pixel or as a linear combination of previously computed MVs in a neighborhood of the current pixel. Note that the negative of the gradient points in the direction of the steepest descent. In (4.50), the first and second terms are prediction and update terms, respectively. The aperture problem is also apparent in the pel-recursive algorithms. Since the update term is a vector along the spatial gradient of image intensity, no correction can be performed in the direction perpendicular to the gradient vector.

The rate of convergence of the Netravali-Robbins algorithm depends on the choice of the step size parameter $\varepsilon$. For example, if $\varepsilon = 1/16$, then at least 32 iterations are required to estimate a displacement by two pixels. On the other hand, a choice of a large step size may cause oscillatory behavior. In order to facilitate faster convergence, an adaptive step size,

$$\varepsilon = \frac{1}{\| \nabla_{\mathbf{x}}\, s_c(\mathbf{x}-\hat{\mathrm{d}}^{(i)};\, t-\Delta t)\, \|^2 + c^2}$$

has been proposed with a bias term $c^2$ to avoid division by zero in areas of constant intensity where the spatial gradient is almost zero. In addition, Walker and Rao [Wal 84] have introduced the heuristic rules: i) If the DFD is less than a threshold, the update term is set equal to zero. ii) If the DFD exceeds a threshold, but the magnitude of the spatial-image gradient is zero, then the update term is again set equal to zero. iii) If the absolute value of the update term (for each MV component) is less than 1/16, then it is set equal to $\pm$ 1/16. iv) If the absolute value of the update term

(for each component) is more than 2, then it is set equal to $\pm 2$. Experimental results indicate that using an adaptive step size improves the convergence of the algorithm.

An alternative method, called Wiener-based motion estimation [Bie 87], computes the least-squares estimate of the update term, given the previous motion vector.

### 4.6.2 Bayesian Motion Estimation

Bayesian methods utilize probabilistic data consistency and motion smoothness constraints, in the form of a probability distribution function (pdf), in order to regularize the motion-estimation problem. Probabilistic smoothness priors can be equivalently expressed in the form of a Markov–Gibbs random field, which models local interactions between motion vectors (see Appendix B). Te objective of Bayesian motion estimation is to maximize the *a posteriori* pdf $p(\mathbf{d}_1, \mathbf{d}_2 \mid \mathbf{s}_k, \mathbf{s}_{k-1})$ of the motion field, where $\mathbf{d}_1, \mathbf{d}_2$ denote the lexicographic ordering of the components of the MV at each pixel, given the observable data, i.e., a pair of image frames $\mathbf{s}_k$ and $\mathbf{s}_{k-1}$. According to the Bayes rule

$$p(\mathbf{d}_1,\mathbf{d}_2 \mid \mathbf{s}_k,\mathbf{s}_{k-1}) = \frac{p(\mathbf{s}_k \mid \mathbf{d}_1,\mathbf{d}_2,\mathbf{s}_{k-1})\, p(\mathbf{d}_1,\mathbf{d}_2 \mid \mathbf{s}_{k-1})}{p(\mathbf{s}_k \mid \mathbf{s}_{k-1})}$$

where the denominator $p(\mathbf{s}_k|\mathbf{s}_{k-1})$ is independent of $(\mathbf{d}_1, \mathbf{d}_2)$, hence it is a scaling constant, and we assume that the *a priori* pdf does not depend on $\mathbf{s}_{k-1}$, i.e., $p(\mathbf{d}_1, \mathbf{d}_2 | \mathbf{s}_{k-1}) = p(\mathbf{d}_1, \mathbf{d}_2)$.

#### Basics of Bayesian Motion Estimation

The conditional pdf $p(\mathbf{s}_k | \mathbf{d}_1, \mathbf{d}_2, \mathbf{s}_{k-1})$ describes the distribution of the current frame $\mathbf{s}_k$ given the reference frame $\mathbf{s}_{k-1}$ and motion vectors. Hence, it models the probability distribution of the displaced frame difference (DFD) (4.26), which is assumed to be a zero-mean Gaussian given by

$$p(\mathbf{s}_k \mid \mathbf{d}_1,\mathbf{d}_2,\mathbf{s}_{k-1}) \approx e^{-\frac{1}{2\sigma_n^2}\sum_{x_1}\sum_{x_2}[s_k(\mathbf{x}) - s_{k-1}(\mathbf{x}+\mathbf{d}(\mathbf{x}))]^2}$$

or the pdf of the error in the optical flow equation (4.23), given by

$$p(\mathbf{s}_k \mid \mathbf{d}_1,\mathbf{d}_2,\mathbf{s}_{k-1}) \approx e^{-\frac{1}{2\sigma_n^2}\sum_{x_1}\sum_{x_2}\left[\frac{\partial s_k(x_1,x_2)}{\partial x_1}d_1(\mathbf{x}) + \frac{\partial s_k(x_1,x_2)}{\partial x_2}d_2(\mathbf{x}) + \frac{\partial s_k(x_1,x_2)}{\partial t}\right]^2}$$

The *a priori* pdf $p(\mathbf{d}_1, \mathbf{d}_2)$ of the motion vector field is defined by a Gibbs distribution [Gem 84],

$$p(\mathbf{d}_1, \mathbf{d}_2) \approx e^{-\frac{\sum_{(\mathbf{x}_i, \mathbf{x}_j) \in C} \|\mathbf{d}(\mathbf{x}_i) - \mathbf{d}(\mathbf{x}_j)\|^2}{T}}$$

which encourages a smooth motion field, where $C$ defines the set of two-pixel cliques within a neighborhood system (see Appendix B) and $T$ is temperature (Appendix C). Since the logarithm is a monotonic function, we can maximize $\ln\{p(\mathbf{s}_k \mid \mathbf{d}_1, \mathbf{d}_2 \mid \mathbf{s}_{k-1})\, p(\mathbf{d}_1, \mathbf{d}_2 \mid \mathbf{s}_{k-1})\}$ or equivalently minimize its negative to find the MAP motion estimates. Then, the MAP estimates $\hat{\mathbf{d}}_1, \hat{\mathbf{d}}_2$ minimize

$$E(\mathbf{d}_1, \mathbf{d}_2) = \frac{1}{2\sigma_n^2} \sum_{x_1} \sum_{x_2} [s_k(\mathbf{x}) - s_{k-1}(\mathbf{x} + \mathbf{d}(\mathbf{x}))]^2 + \frac{1}{T} \sum_{c \in C} \| \mathbf{d}(\mathbf{x}_i) - \mathbf{d}(\mathbf{x}_j) \|^2 \quad (4.44)$$

and the Bayesian motion-estimation problem reduces to an energy minimization problem. Observe that the energy function (4.44) is similar to (4.34), which is used in the Horn–Schunck method, in that both contain a data consistency term and a smoothness term. The main difference is that this is a nonlinear optimization problem with possibly many local minima. Simulated annealing, greedy methods, or fast primal-dual (Fast-PD) optimization [Glo 08] have been used to minimize (4.44). Hierarchical Bayesian motion-estimation formulations have been shown to yield a more regular energy function with fewer local minima.

### Bayesian Motion Estimation with Discontinuity Modeling

The basic Bayesian motion-estimation formulation imposes data consistency and global smoothness constraints across the entire image, hence failing to deal with motion boundaries and occlusion areas properly. The line field has been introduced into the Bayesian framework to model motion boundaries [Kon 92]. The maximum *a posteriori* probability (MAP) estimate of the MV field $(\mathbf{d}_1, \mathbf{d}_2)$ and the line field $\mathbf{l}$ is defined by

$$(\hat{\mathbf{d}}_1, \hat{\mathbf{d}}_2, \hat{\mathbf{l}}) = \arg\max_{\mathbf{d}_1, \mathbf{d}_2, \mathbf{l}}\, p(\mathbf{d}_1, \mathbf{d}_2, \mathbf{l} \mid \mathbf{s}_k, \mathbf{s}_{k-1}) \quad (4.45)$$

where $\mathbf{l}$ is a binary line (segmentation) field that models discontinuities in the MV field, where $l_{ij} = 1$ indicates a motion border is present between sites $x_i$ and $x_j$. Hence, we now have three unknowns per pixel, $d_1, d_2$ and $l$. Using the Bayes theorem,

$$p(\mathbf{d}_1, \mathbf{d}_2, \mathbf{l} \mid \mathbf{s}_k, \mathbf{s}_{k-1}) = \frac{p(\mathbf{s}_k \mid \mathbf{d}_1, \mathbf{d}_2, \mathbf{l}, \mathbf{s}_{k-1})\, p(\mathbf{d}_1, \mathbf{d}_2 \mid \mathbf{l}, \mathbf{s}_{k-1})\, p(\mathbf{l} \mid \mathbf{s}_{k-1})}{p(\mathbf{s}_k \mid \mathbf{s}_{k-1})}$$

where $p(\mathbf{s}_k | \mathbf{d}_1, \mathbf{d}_2, \mathbf{l}, \mathbf{s}_{k-1})$, $p(\mathbf{d}_1, \mathbf{d}_2 | \mathbf{l}, \mathbf{s}_{k-1})$, and $p(\mathbf{l} | \mathbf{s}_{k-1})$ are modeled by Gibbs distributions with potentials $U_s(\mathbf{s}_k | \mathbf{d}_1, \mathbf{d}_2 \mathbf{l}, \mathbf{s}_{k-1})$, $U_d(\mathbf{d}_1, \mathbf{d}_2 | \mathbf{l}, \mathbf{s}_{k-1})$, and $U_l(\mathbf{l} | \mathbf{s}_{k-1})$, respectively. The first potential is a likelihood function derived from a Gaussian pdf, while the latter two are Gibbs priors that can expressed as the sum of clique potentials to impose smoothness constraints.

The MAP estimate of the motion field and associated line field can be computed by minimizing a cost function that consists of an optical flow error (first term) and a Gibbs potential (second and third terms) (penalizing discontinuities in the estimated motion field) [Kon 92] as

$$(\hat{\mathbf{d}}_1, \hat{\mathbf{d}}_2, \hat{\mathbf{l}}) = \arg\min_{\mathbf{d}_1, \mathbf{d}_2, \mathbf{l}} U_s(\mathbf{s}_k \mid \mathbf{d}_1, \mathbf{d}_2, \mathbf{l}, \mathbf{s}_{k-1}) + \lambda_d U_d(\mathbf{d}_1, \mathbf{d}_2 \mid \mathbf{l}, \mathbf{s}_{k-1}) + \lambda_l U_l(\mathbf{l} \mid \mathbf{s}_{k-1})$$

This energy minimization problem can be solved by a simulated annealing procedure or a deterministic approximation of it. The main drawback of Bayesian methods is that simulated annealing procedures require an extensive amount of computation, whereas deterministic procedures may be caught in a local minimum of the cost function. An occlusion field has also been added to the formulation [Dub 93]. It has been shown that extensions of the Horn–Schunck solution using directional smoothness constraints can be seen as a special case (using deterministic constraints) of this Bayesian formulation.

## 4.7 Transform-Domain Methods

Transform-domain methods can be classified as phase-correlation methods [Kug 75, For 02] and space-frequency spectral methods [Hee 88, Bar 94].

### 4.7.1 Phase-Correlation Method

The phase-correlation method, first proposed by [Kug 75], exploits the fact that translation in the image domain corresponds to a linear phase-shift in the 2D spatial frequency domain, since $s_{k+1}(\mathbf{x}) = s_k(\mathbf{x} + \mathbf{d}(\mathbf{x}))$ implies

$$S_{k+1}(f_1, f_2) = e^{-j(d_1 f_1 + d_2 f_2)} \, S_k(f_1, f_2) \tag{4.46}$$

The linear term of the Fourier phase difference between two frames determines the motion estimate. The phase-correlation function is given by

$$C_{k,k+1}(f_1, f_2) = \frac{S_{k+1}(f_1, f_2) S_k^*(f_1, f_2)}{\left| S_{k+1}(f_1, f_2) S_k^*(f_1, f_2) \right|} \tag{4.47}$$

where $S_k(f_1, f_2)$ is the 2D Fourier transform of the frame $k$ with respect to the spatial variables $x_1$ and $x_2$, and * denotes complex conjugation. The method is insensitive to frame-to-frame intensity shifts (bias or multiplication by a constant), since they do not affect the Fourier phase.

If there is a translational motion between frames $k$ and $k+1$, $C_{k,k+1}(f_1, f_2) = e^{-j(d_1 f_1 + d_2 f_2)}$ and the inverse 2D Fourier transform of (4.47) yields

$$c_{k,k+1}[n_1, n_2] = \delta[n_1 - d_1, n_2 - d_2] \tag{4.48}$$

which is an impulse whose location indicates the displacement vector $(d_1, d_2)$.

### Maximum-Displacement Estimate/Block Size

Since the DFT is periodic by the block size $N_1 \times N_2$, and the DFT of real images exhibit Hermitian symmetry, the maximum range of displacement estimates is limited to $[-(N_i/2) + 1, N_i/2]$ for $N_i$ even. For example, to estimate displacements within a range [-31,32], the block size should be at least $64 \times 64$.

### Boundary Effects

In order to obtain a perfect impulse in the inverse 2D-DFT, the shift must be cyclic for each block. Since things disappearing at one end of a block do not generally reappear at the other end, the impulse will degenerate into a peak due to windowing with a rectangular kernel.

### Multiple Motions

Experiments indicate that multiple peaks are observed if there are multiple motions. An additional search is required to find which peak belongs to which part of the block.

### 4.7.2 Space-Frequency Spectral Methods

Space-frequency spectral methods exploit the fact that Fourier power spectrum of a translating image lies on a plane through the origin of the 3D spatio-temporal frequency domain. Thus, energy-based methods compute translational velocity by comparing the output energy of a set of velocity-tuned filters [Hee 88, Fle 90, Bar 94].

## 4.8 3D Motion and Structure Estimation

3D-motion/pose and structure (shape) estimation methods can be broadly classified as structure from motion (SFM) and structure from stereo (SFS) methods. SFM methods can be further classified as *sparse structure* from a set of feature correspondences and *direct methods* that estimate dense structure from intensity gradients (without explicit correspondence estimation). SFS (or from multiple views) is usually preferred for dense structure (or depth map) estimation if stereo or multi-view video is available. These methods are applicable to both uncalibrated (for projective or affine reconstruction) or calibrated (for Euclidean reconstruction) cameras.

Early works on SFM and SFS considered Euclidean reconstruction using calibrated cameras only [Adi 85, Agg 88]. Given two calibrated cameras, their relative orientations can be determined from the epipolar constraint represented algebraically by the "essential matrix" $\mathbf{E}$, which depends on the rotation $\mathbf{R}$ and translation $\mathbf{t}$ between the two cameras. Once the essential matrix is estimated from at least eight image-point correspondences, the 3D scene structure can be recovered (see (4.52)) relative to the coordinate frame of a reference camera (reference frame).

Koenderink and Van Doorn [Koe 90] were first to propose solving the SFM problem (for orthographic cameras) in two phases: i) first, reconstruct a unique (up to an arbitrary affine transformation) 3D scene representation, called the affine structure, from at least two views without calibration; ii) then, use available metric measurements (distances or angles) to uniquely determine the Euclidean structure. Faugeras [Fau 95] extended this approach to projective reconstruction, which is unique up to a projective transformation, from uncalibrated cameras, and proposed the stratification of the 3D reconstruction methods into projective, affine, and Euclidean stages, where projective or affine reconstructions may suffice for some robot vision and synthetic view synthesis applications without going through a laborious calibration process needed for Euclidean reconstruction. When calibration is not available, the epipolar geometry is represented by the "fundamental matrix," which also incorporates unknown camera calibration information. Projective and affine reconstructions

compute 3D coordinates of a sparse set of points relative to reference frames defined by five and four selected points in projective and affine spaces, respectively. In projective and affine reconstructions, the unknown camera calibration matrices, where intrinsic and extrinsic camera parameters are allowed to vary from frame to frame, are folded into the overall projective or affine deformation (ambiguity) of the solution with respect to the Euclidean solution.

Literature on 3D motion and structure estimation is extensive; there are many papers, book chapters [Zha 04], and complete books [Har 04] on the subject. Our coverage here will be at the beginner level to introduce the main concepts and popular methods. We start with the basics of camera calibration in Section 4.8.1. Sparse affine reconstruction using an uncalibrated camera is covered in Section 4.8.2. Section 4.8.3 treats uncalibrated sparse projective reconstruction. Euclidean reconstruction using full or partial calibration is discussed in Section 4.8.4. We introduce a direct method for dense planar parallax estimation in Section 4.8.5. Finally, Section 4.8.6 covers dense structure estimation from stereo (multi-view) images/video.

## 4.8.1 Camera Calibration

Camera calibration is an essential step to estimate metric (Euclidean) structure from video. It refers to estimation of intrinsic and extrinsic camera parameters (a total of 11 free parameters) in the camera model (4.3). The intrinsic camera matrix $\mathbf{K}$ has 5 degrees of freedom, where $(x_{1,0}, x_{2,0})$ denotes center of the image, $s$ is the skew parameter, $f$ is the focal length of the camera, and $k_1/k_2$ denote the aspect ratio. The extrinsic camera parameters are the rotation matrix $\mathbf{R}$ with three degrees of freedom and the translation vector $\mathbf{t}$ with three parameters, which model the rotation and translation of the camera coordinate system with respect to the scene (world) coordinate system, respectively.

Camera-calibration techniques can be classsified as pre-calibration methods using known reference objects and auto-calibration (or self-calibration) methods that only rely on point correspondences from an actual video scene without any reference objects. Pre-calibration methods can utilize multiple images of a known 3D reference object [Tsa 87] or a planar target [Zha 00] from different viewpoints. Pre-calibration techniques can be classified as direct estimation of parameters vs. two-step estimation: first estimation of the camera projection matrix $\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}]$ and then recovery of the intrinsic and extrinsic parameters from entries of the projection matrix. Pre-calibration should be preferred whenever possible, since self-calibration cannot always achieve the same level of accuracy as that of pre-calibration [Zha 04].

In some cases, it is only possible to perform partial pre-calibration to determine the camera center, pixel aspect ratio, and skewness parameter, since the focal length and camera rotation and translation may vary during the actual recording (capture). A popular method for pre-calibration using a 3D reference object consists of four steps:

1. Detect image feature points $\mathbf{x}_i$ corresponding to 3D reference object points with known world coordinates $\mathbf{X}_i$;
2. Estimate the camera projection matrix $\mathbf{P}$ from at least six world-image point correspondence pairs by solving $\mathbf{x} = \mathbf{PX}$ using the method of linear least squares (Appendix D);
3. Compute the intrinsic and extrinsic camera parameters $\mathbf{K}$, $\mathbf{R}$ and $\mathbf{t}$ as closed-form functions of entries of matrix $\mathbf{P}$ [Zha 04];
4. Refine $\mathbf{K}$, $\mathbf{R}$, and $\mathbf{t}$ through nonlinear optimization of

$$\min_{\mathbf{K},\mathbf{R}\,\mathbf{t}} \sum_i \| \mathbf{x}_i - \mathbf{P}\ \mathbf{X}_i \|^2$$

starting with the estimates in step 3. Alternatively, we can first refine $\mathbf{P}$ through nonlinear optimization (i.e., complete step 4 right after step 2) and then determine $\mathbf{K}$, $\mathbf{R}$, and $\mathbf{t}$ from refined $\mathbf{P}$.

A similar procedure to recover intrinsic and extrinsic camera parameters from a homography estimated by using eight world-image point-correspondence pairs has been proposed by Zhang [Zha 00, Zha 04] when a simpler planar (2D) test pattern is used.

### 4.8.2 Affine Reconstruction

Recall from Section 4.1.1 that the affine camera model (4.7) covers orthographic, weak-perspective, and paraperspective projection models and provides a reasonable approximation for imaging of distant or limited depth scenes. The affine structure reconstruction problem can be defined as [Koe 90]: Given at least five point correspondences between two views captured by an uncalibrated affine camera, defined by (4.7), arbitrarily choose four of these points that are not in a degenerate configuration, and reconstruct the fifth point (and any other points with known correspondences) in the affine coordinate system defined by these four points. The affine structure differs from the Euclidean structure by an unknown affine transformation, which alters metric distances and angles, but preserves parallelism. This transformation can be

computed in the second step of stratification using camera-calibration parameters or some metric measurements about the scene to recover the Euclidean structure up to a scale factor. Koenderink-Van Doorn [Koe 90] offered a geometric solution to the two-view affine structure reconstruction problem.

### Multi-View Affine Reconstruction – Factorization Method

Often we have more than two views of a scene and would like to estimate all camera matrices and affine structure of features at once. Tomasi–Kanade [Tom 92] proposed an elegant algebraic solution to this multi-view problem. Let's assume we have correspondence information between $N$ feature points over $M$ views denoted by $\mathbf{x}_{ij}$, $i = 1, \ldots, M, j = 1, \ldots, N$. They made the observation that in the case of an affine camera and rigid motion, $2M \times N$ measurement matrix $\mathbf{W}$, which stacks 2D image coordinates of all corresponding points in successive views, has rank 3. This reduced rank of the measurement matrix $\mathbf{W}$ comes from the fact that position of feature points in the image plane is constrained by the rigid 3D motion. Furthermore, given Eqn. (4.7), it can be expressed as product of camera-motion matrix $\mathcal{M}$ and affine-structure matrix $\mathcal{X}$ as

$$\begin{aligned} \mathbf{W} &= \begin{bmatrix} \mathbf{x}_{11} & \mathbf{x}_{12} & \cdots & \mathbf{x}_{1N} \\ \mathbf{x}_{21} & \mathbf{x}_{22} & \cdots & \mathbf{x}_{2N} \\ \vdots & & \ddots & \vdots \\ \mathbf{x}_{M1} & \mathbf{x}_{M2} & \cdots & \mathbf{x}_{MN} \end{bmatrix}_{2M \times N} \\ &= \begin{bmatrix} \mathbf{M}_1 \\ \mathbf{M}_2 \\ \vdots \\ \mathbf{M}_M \end{bmatrix}_{2M \times 3} [\mathbf{X}_1 \quad \mathbf{X}_2 \quad \cdots \quad \mathbf{X}_N]_{3 \times N} = \mathcal{M}\mathcal{X} \end{aligned} \tag{4.49}$$

where each camera matrix $\mathbf{M}_i$ is $2 \times 3$, $\mathbf{x}_{ij}$ is $2 \times 1$, and $\mathbf{X}_i$ is $3 \times 1$. Tomasi and Kanade [Tom 92] proposed to factorize the measurement matrix $\mathbf{W}$ using the singular-value decomposition (SVD)

$$\mathbf{W} = \mathbf{U}\mathbf{D}\mathbf{V}^T \tag{4.50}$$

where $\mathcal{M} = \mathbf{U}$ corresponds to motion (camera) matrices and $\mathcal{X} = \mathbf{D}\mathbf{V}^T$ corresponds to a matrix formed by affine-structure vectors relative to a coordinate system centered at one of these points. Hence, the multi-view affine-reconstruction method

can be summarized as: i) form the measurement matrix $\mathbf{W}$, and ii) perform the SVD decomposition of $\mathbf{W}$ to recover $\mathcal{M}$ and $\mathcal{X}$.

### 4.8.3 Projective Reconstruction

When 3D scene reconstruction is based on views recorded by uncalibrated perspective camera(s), the resulting scene structure, called projective reconstruction, differs from the Euclidean geometry by an unknown projective transformation. This is due to the ambiguity stated by (4.7) that any pair of camera matrices $\mathbf{P}_i\mathbf{H}$ and structure matrices $\mathbf{H}^{-1}\mathbf{X}_j$ yield the same projected image coordinates and are projectively equivalent. This unknown projective transformation can be recovered in the second stage (see Section 4.8.4) from camera-calibration information up to a scale factor. Here, we discuss projective reconstruction from two views and multiple views.

#### Two-View Projective Reconstruction – Epipolar Geometry

3D reconstruction of a point $\mathbf{X}_j$ given its image-plane coordinates $\mathbf{x}_{1j}$ and $\mathbf{x}_{2j}$ on two views is based on *epipolar geometry*, which states the lines joining $\mathbf{x}_{1j}$ with its camera center $\mathbf{O}_1$ and $\mathbf{x}_{2j}$ with its camera center $\mathbf{O}_2$ intersect at the point $\mathbf{X}_j$, or alternatively the line from $\mathbf{O}_1$ to $\mathbf{x}_{1j}$, the line from $\mathbf{O}_2$ to $\mathbf{x}_{2j}$, and the line from $\mathbf{O}_1$ to $\mathbf{O}_2$ are all co-planar as depicted in Figure 4.21.

Fundamental matrix $\mathbf{F}$ captures this epipolar geometry between two views, which only depends on camera parameters and pose, in an algebraic expression in the homogeneous coordinates, given by

$$\mathbf{x}_{1j}^{\mathrm{T}}\mathbf{F}\mathbf{x}_{2j} = 0,\ j = 1, \ldots, N \tag{4.51}$$

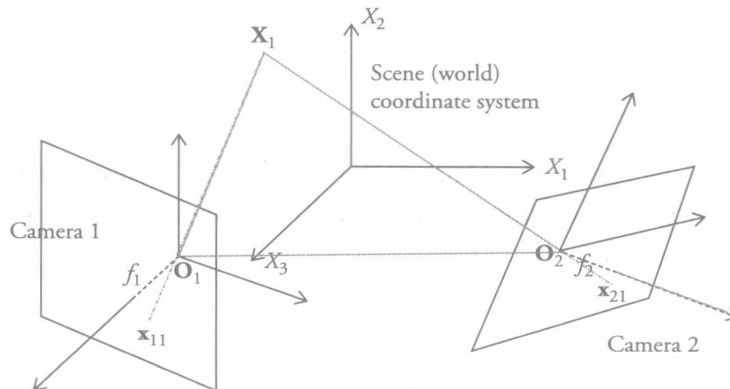



**Figure 4.21** Two-view epipolar geometry between converging projective cameras.

It can be shown that the fundamental matrix can be expressed in terms of the intrinsic and extrinsic camera parameters as $\mathbf{F} = \left(\mathbf{K}_1^{-1}\right)^{\mathrm{T}} \mathbf{E}\,\mathbf{K}_2^{-1}$, where $\mathbf{E} = \mathbf{t}_\times \mathbf{R}$ is called the essential matrix and

$$\mathbf{t}_\times = \begin{bmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix}$$

When we do not have access to the internal calibration parameters $\mathbf{K}_1$ and $\mathbf{K}_2$, the reconstruction obtained from $\mathbf{F}$ will be up to an unknown projective transformation $\mathbf{H}^{-1}$ of the actual Euclidean structure. A complete algorithm for two-view reconstruction of the 3D projective structure can be summarized as:

1. *Pixel correspondences:* Find at least $N = 8$ pixel correspondences between two views.
2. *Normalization* [Har 97]: Normalize the pixel coordinates $\tilde{\mathbf{x}}_{ij} = \mathbf{T}_i \mathbf{x}_{ij}$, $i = 1,2$, $j = 1, \ldots, N$, where affine mappings $\mathbf{T}_i$ are computed as follows:
   a. Shift the origin of the coordinates to center at the mean $\bar{\mathbf{x}}_i = \sum_{j=1}^{N} \mathbf{x}_{ij}$, $i = 1,2$.
   b. Scale the shifted pixels, $\tilde{\mathbf{x}}_{ij} = k(\mathbf{x}_{ij} - \bar{\mathbf{x}}_i)$ such that their root mean-squared distance from the origin is $\sqrt{2}$.
3. *Fundamental matrix:* Given normalized correspondences, estimate the fundamental matrix $\tilde{\mathbf{F}}$.
   a. Let $\tilde{x}_{ij} = [u_{ij}\ v_{ij}\ w_{ij}]$, $i = 1,2$, $j = 1,\ldots,N$; set up an equation $\mathbf{Af} = 0$, where $\mathbf{f} = [\tilde{F}_{11}\ \tilde{F}_{12} \ldots \tilde{F}_{33}]^T$ is a $9 \times 1$ vector containing entries of the $3 \times 3$ fundamental matrix $\tilde{\mathbf{F}}$, and row $j$ of the $N \times 9$ matrix $\mathbf{A}$ is of the form

$$[u_{2j}u_{1j} \quad u_{2j}v_{1j} \quad u_{2j}w_{1j} \quad v_{2j}u_{1j} \quad v_{2j}v_{1j} \quad v_{2j}w_{1j} \quad w_{2j}u_{1j} \quad w_{2j}v_{1j} \quad w_{2j}w_{1j}]$$

   We note that the rank of matrix $\mathbf{A}$ must be 8 for a unique solution to exist.

   b. The non-trivial solution to this set of homogeneous equations can be found by solving

$$\min \|\mathbf{Af}\|$$
$$\text{subject to } \|\mathbf{f}\| = 1$$

The solution is the eigenvector of $\mathbf{A}^T\mathbf{A}$ corresponding to the smallest eigenvalue (see Appendix D).

c. The fundamental matrix $\mathbf{F}$ before normalization is given by $\mathbf{F} = \mathbf{T}_2^{\mathrm{T}}\tilde{\mathbf{F}}\mathbf{T}_1$.

4. *Camera projection matrices:* Given $\mathbf{F}$, recover camera matrices $\mathbf{P}_1 = [\mathbf{I}\ \mathbf{0}]$ and $\mathbf{P}_2 = [\mathbf{R}\ \mathbf{t}]$.

   a. Compute the singular value decomposition $\mathbf{F} = \mathbf{UDV}^{\mathrm{T}}$. Since $\mathbf{F}$ has rank 2, it should have two non-zero singular values $\mathbf{D} \approx \mathrm{diag}(a, b, 0)$ in the absence of noise.

   b. Observe that [Har 04]

$$\mathbf{F} = (\mathbf{UZU}^{\mathrm{T}})(\mathbf{UY}^{\mathrm{T}}\hat{\mathbf{D}}\mathbf{V}^{\mathrm{T}}) = \mathbf{S}\ \mathbf{M},$$

   where

$$\mathbf{Y} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \mathbf{Z} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

   and

$$\hat{\mathbf{D}} \approx \mathrm{diag}(a, b, c)$$

   The value of $c$ can be set arbitrarily; a common choice is $c = (a + b)/2$.

   c. Then, the camera matrices are $\mathbf{P}_1 = [\mathbf{I}\ \mathbf{0}]$ and $\mathbf{P}_2 = [\mathbf{M}\ \mathbf{u}_3]$, where $\mathbf{u}_3$ is the third column of $\mathbf{U}$. Note that $\mathbf{u}_3^{\mathrm{T}}\mathbf{F} = 0$, i.e., $\mathbf{u}_3$ is the generator of the left null-space of $\mathbf{F}$.

5. *Triangulation:* The lines joining $\mathbf{x}_{1j}$ and $\mathbf{x}_{2j}$ with their respective camera centers may not intersect in 3D due to estimation inaccuracies, hence we estimate 3D points $\hat{\mathbf{X}}_j$ by solving

$$\mathrm{Min}_{\hat{\mathbf{X}}_j} \| \mathbf{x}_{1j} - \mathbf{P}_1\hat{\mathbf{X}}_j \|^2 + \| \mathbf{x}_{2j} - \mathbf{P}_2\hat{\mathbf{X}}_j \|^2$$

   which is robust to noise [Har 04]. This step requires an iterative optimization procedure.

We discuss the case of 3D Euclidean reconstruction when the cameras are calibrated, i.e., $\mathbf{K}_1$ and $\mathbf{K}_2$ are known, in Section 4.8.4.

### Multi-View Reconstruction – Projective Factorization

Often we have more than two views of a scene and would like to recover all camera matrices and 3D structure of feature points at once. To this effect, Tomasi–Kanade factorization for affine reconstruction has been extended for projective reconstruction by Sturm–Triggs [Stu 96] and others [Ole 07]. The projection equations (4.6) for $M$ views and $N$ feature points can be written in vector-matrix form as

$$\begin{aligned} \boldsymbol{O} &= \begin{bmatrix} \lambda_{11}\mathbf{x}_{11} & \lambda_{12}\mathbf{x}_{12} & \cdots & \lambda_{1N}\mathbf{x}_{1N} \\ \lambda_{21}\mathbf{x}_{21} & \lambda_{22}\mathbf{x}_{22} & \cdots & \lambda_{2N}\mathbf{x}_{2N} \\ \vdots & & \ddots & \vdots \\ \lambda_{M1}\mathbf{x}_{M1} & \lambda_{M2}\mathbf{x}_{M2} & \cdots & \lambda_{MN}\mathbf{x}_{MN} \end{bmatrix}_{3M\times N} \\ &= \begin{bmatrix} \mathbf{P}_1 \\ \mathbf{P}_2 \\ \vdots \\ \mathbf{P}_M \end{bmatrix}_{3M\times 4} \begin{bmatrix} \mathbf{X}_1 & \mathbf{X}_2 & \cdots & \mathbf{X}_N \end{bmatrix}_{4\times N} \end{aligned} \tag{4.52}$$

where the scale factors $\lambda_{ij}$ are called projective depths. Note that it is required that every point is visible in every view (no occlusions), so the observation matrix $\boldsymbol{O}$ has no missing entries. We can express the matrix $\boldsymbol{O}$ as Hadamard (element-wise) product $\odot$ of the matrices

$$\boldsymbol{O} = \boldsymbol{\Lambda} \odot \mathbf{W}, \quad \text{where } \boldsymbol{\Lambda} = \begin{bmatrix} \lambda_{11} & \lambda_{12} & \cdots & \lambda_{1N} \\ \lambda_{21} & \lambda_{22} & \cdots & \lambda_{2N} \\ \vdots & & \ddots & \vdots \\ \lambda_{M1} & \lambda_{M2} & \cdots & \lambda_{MN} \end{bmatrix} \text{ and}$$

$$\mathbf{W} = \begin{bmatrix} \mathbf{x}_{11} & \mathbf{x}_{12} & \cdots & \mathbf{x}_{1N} \\ \mathbf{x}_{21} & \mathbf{x}_{22} & \cdots & \mathbf{x}_{2N} \\ \vdots & & \ddots & \vdots \\ \mathbf{x}_{M1} & \mathbf{x}_{M2} & \cdots & \mathbf{x}_{MN} \end{bmatrix}$$

The projective factorization aims to solve for the unknown projective depths $\Lambda$, the unknown cameras $\mathbf{P}$, and the unknown structure $\mathbf{X}$ simultaneously. Inspection of the right-hand side of (4.52) reveals that the matrix $\boldsymbol{O}$ must have rank 4. However, since the projected depths $\lambda_{ij}$ are unknown, this rank-4 property is not satisfied if

they are set arbitrarily; hence, we need an accurate initialization of projected depths. Sturm–Triggs [Stu 96] proposed an iterative algorithm to solve the problem by alternation: i) given $\mathbf{\Lambda}$, solve for $\mathbf{P}$ and $\mathbf{X}$ by SVD factorization; ii) given $\mathbf{P}$ and $\mathbf{X}$, solve for $\mathbf{\Lambda}$ by least-squares fitting; and alternate between these two steps until convergence. In their original paper, Sturm–Triggs initialize projective depths from successive two-view reconstructions. Several generalizations of the original method share the following steps:

1. Form the homogeneous observation matrix $\mathbf{W}$. Initialize $\mathbf{\Lambda}^{(0)}$, such that all $\lambda_{ij}^{(0)} = 1$.
2. Repeat for $k = 0, \ldots, N$.
   a. Compute the closest rank-4 matrix $\boldsymbol{O}^{(k)}$ to $\Lambda^{(k)} \odot \mathbf{W}$ by singular value decomposition. Let $\mathbf{\Lambda}^{(k)} \odot \mathbf{W} = \mathbf{U}\mathbf{D}\mathbf{V}^{\mathrm{T}}$. Define $\hat{\mathbf{D}}$ to be the diagonal matrix obtained by keeping the first four (largest) diagonal entries of $\mathbf{D}$ and setting the rest equal to zero. Then, $\boldsymbol{O}^{(k)} = \mathbf{U}\hat{\mathbf{D}}\mathbf{V}^{\mathrm{T}}$.
   b. If $k = N$ go to step 3. Otherwise, compute new matrix $\mathbf{\Lambda}^{(k+1)}$ of weights $\lambda_{ij}^{(k+1)}$ so that $\Lambda^{(k+1)} \odot \mathbf{W}$ is as close as possible to $\boldsymbol{O}^{(k)}$ under Frobenius norm.
3. Compute the factorization $\boldsymbol{O}^{(N)} = \mathbf{P}\,\mathbf{X}$ to obtain $\mathbf{P}$ and $\mathbf{X}$ that provide the camera matrices and point locations, respectively.

It has been noted that the Sturm–Triggs method with the above initialization works well if the true solution is close to its affine approximation [Har 04]. Dai *et al.* [Dai 10] proposed a non-iterative element-wise factorization that can also handle missing data in the measurement matrix.

### Bundle Adjustment

Since the observed feature-point correspondences $\mathbf{x}_{ij}$ in the two-view and multi-view problems have limited accuracy, the projection equations $\lambda_{ij}\mathbf{x}_{ij} = \mathbf{P}_i\mathbf{X}_j$ cannot be satisfied exactly. Therefore, projection matrices $\mathbf{P}_i$ and 3D points $\mathbf{X}_j$ estimated by using simple linear estimation procedures discussed above may contain arbitrary errors. The maximum likelihood (ML) estimation of $\mathbf{P}_i$ and $\mathbf{X}_j$, under the assumption that measurement errors are Gaussian and independent of each other, requires the solution of the following nonlinear least-squares problem:

$$\min_{\mathbf{P}_i, \mathbf{X}_j} \sum_i \sum_{j=1}^{N} \left\| \mathbf{x}_{ij} - \mathbf{P}_i\,\mathbf{X}_j \right\|^2 \tag{4.53}$$

which is generally referred as the "bundle adjustment" method. A popular iterative algorithm to minimize this cost function is the Levenberg–Marquardt method [Tri 99, Har 04]. In order to converge to the globally optimal solution, it requires a good initial solution, which can be found by applying the above two-view or multi-view solutions.

### 4.8.4 Euclidean Reconstruction

The classical approach to Euclidean reconstruction requires knowledge of the camera-calibration matrices. In the two-view case, given the calibration matrices $\mathbf{K}_1$ and $\mathbf{K}_2$, we can first compute

$$\breve{x}_{ij} = \mathbf{K}_{\mathrm{i}}^{-1} x_{ij},\ i = 1,2 \tag{4.54a}$$

Then, Eqn. (4.51) can be rewritten as $\mathbf{x}_{1j}^{\mathrm{T}}(\mathbf{K}_1^{-1})^{\mathrm{T}}\mathbf{E}\mathbf{K}_2^{-1}\mathbf{x}_{2j} = 0$, which can be re-expressed as

$$\breve{x}_{1j}^{T}\mathbf{E}\breve{x}_{2j} = 0,\ j = 1, \ldots, N \tag{4.54b}$$

Hence, the essential matrix $\mathbf{E}$ and the 3D Euclidean reconstruction (up to a global scale) can be recovered from the scaled pixel correspondences (4.54a) directly using the procedure described in Section 4.8.3.

Alternatively, in the stratification approach [Fau 95], a projective reconstruction is first computed from uncalibrated pixel correspondences, and is then upgraded to an affine or a Euclidean one by incorporating ground-truth or camera-calibration information. This can be implemented in different ways given different information:

1. *Camera calibration:* Projective ambiguity can be resolved up to a global-scale factor given the camera matrices.
2. *Partial camera calibration:* Constraints, such as known focal length, or that cameras have the same internal parameters, may be enforced during bundle-adjustment.
3. *Auto-calibration:* Automatic methods for auto-calibration often compute an affine reconstruction first, followed by an update to a Euclidean reconstruction and full determination of the camera-calibration parameters [Har 04].
4. *Ground-truth 3D coordinates:* Knowledge of 3D Euclidean coordinates of at least five ground-control points is required [Har 04].

### 4.8.5 Planar-Parallax and Relative Affine Structure Reconstruction

The planar-parallax framework, introduced in Section 4.2.2, enables reconstruction of a dense scene structure (height of pixels) relative to a reference plane in the scene from two or multiple frames. First, a planar surface that is visible in all views is identified. Then, homography between each view and a reference view is estimated successively (Section 4.5.4), and all views are registered with respect to the reference view to compute the residual (planar parallax) motion fields at each frame. Here, we discuss direct estimation of the unknown motion parameters, which are the epipoles (one for each frame) and dense structure parameters, i.e., the height of each pixel with respect to reference plane at every frame, in the residual motion model (4.7) from uncalibrated cameras. The iterative direct-estimation method is based on the "the epipolar brightness constraint" [Ira 02], which is obtained by substituting the components of the planar-parallax flow (4.7) (assuming the time interval between two frames is one)

$$
\begin{aligned}
v_1 &= \hat{x}_{1,w} - x_1 = -\frac{\gamma(\mathbf{x},k)}{\left(1+\gamma(\mathbf{x},k)\right)t_{3k}}\left(t_{3k}x_1 - t_{1k}\right) \\
v_2 &= \hat{x}_{2,w} - x_2 = -\frac{\gamma(\mathbf{x},k)}{\left(1+\gamma(\mathbf{x},k)\right)t_{3k}}\left(t_{3k}x_2 - t_{2k}\right)
\end{aligned}
$$

into the optical flow equation (4.24), relating frame $s(\mathbf{x}, k)$ to the reference frame $s_0(\mathbf{x})$,

$$
\frac{\gamma(\mathbf{x},k)}{(1+\gamma(\mathbf{x},k))t_{3k}}\left\{\frac{\partial s_0(\mathbf{x})}{\partial x_1}(t_{3k}x_1 - t_{1k}) + \frac{\partial s_0(\mathbf{x})}{\partial x_2}(t_{3k}x_2 - t_{2k})\right\} + \Delta_t(s(\mathbf{x},k)) = 0 \quad (4.55)
$$

where

$$
\Delta_t(s(\mathbf{x},k)) = s(x_1 - \hat{x}_{1,w}, x_2 - \hat{x}_{2,w}, k) - s_0(x_1, x_2) - \frac{\partial s_0(\mathbf{x})}{\partial x_1}\hat{x}_{1,w} - \frac{\partial s_0(\mathbf{x})}{\partial x_2}\hat{x}_{2,w}
$$

We have one constraint equation for each pixel for a total of $LN$ equations, but there are total of $L(N+3)$ unknowns, where $L$ and $N$ denote the number of frames

and number of pixels per frame, respectively. Hence, we require a local smoothness constraint on the shape parameters, whereby $\gamma$ is assumed to be constant within a local ($5 \times 5$) window $\boldsymbol{B}$ (see step 3a below) about each pixel, so that the problem is not under-constrained. The complete algorithm can be summarized as follows [Ira 02]:

1. Construct a multi-resolution pyramid for each frame $s(\mathbf{x}, k)$ and the reference frame $s_0(\mathbf{x})$.
2. Start with the coarsest resolution level. Initialize structure parameters $\gamma^{(0)}(x_1, x_2, k) = 0$ for all $(x_1, x_2)$ at each frame $k$ and the motion parameter $\mathbf{t}_k^{(0)} = [0\ 0\ 1]^{\mathrm{T}}$ for each frame $k$.
3. Refine structure and motion parameters iteratively at the current resolution level. Set $j = 1$.

   a. For all frames $k$, given epipole $\mathbf{t}_k^{(j)}$, estimate the structure for each pixel $(\bar{x}_1, \bar{x}_2)$ by minimizing:

$$E(\boldsymbol{\gamma}^{(j)}(\bar{x}_1, \bar{x}_2)) =$$

$$\sum_k \sum_{(x_1,x_2)\in B(\bar{x}_1,\bar{x}_2)} \left[\gamma\left\{\frac{\partial s_0}{\partial x_1}(t_{3k}x_1 - t_{1k}) + \frac{\partial s_0}{\partial x_2}(t_{3k}x_2 - t_{2k})\right\} + (1+\gamma)t_{3k}\Delta_t(s(\mathbf{x},k))\right]^2$$

   b. Given the scene structure parameters for all pixels, estimate the position of the epipole for each frame with respect to the reference frame. For each epipole $\boldsymbol{t}_k$, minimize

$$E(\mathbf{t}_k^{(j)}) =$$

$$\sum_{(x_1,x_2)} \left\{W_k\left[\gamma\left\{\frac{\partial s_0}{\partial x_1}(t_{3k}x_1 - t_{1k}) + \frac{\partial s_0}{\partial x_2}(t_{3k}x_2 - t_{2k})\right\} + (1+\gamma)t_{3k}\Delta_t(s(\mathbf{x},k))\right]\right\}^2$$

   where $W_k, \gamma$, and partials are functions of $(x_1, x_2)$ and the weights $W_k = 1/[(1+\gamma)t_{3k}]$.

   c. Set $j = j + 1$, and repeat (a) and (b) five times at each resolution level.
4. Repeat step 3 at the next higher resolution level (until the original resolution) using parameter estimates from the previous level as initial estimates.
5. The final output is the structure and motion parameters at the original resolution of input views and the residual parallax-flow field $(v_1, v_2)$ synthesized from them using (4.7).

The risk of getting stuck at a local minimum is significantly reduced by using a coarse-to-fine estimation strategy and the limited search space (*three* parameters per frame for global-motion estimation and *one* parameter per pixel for local correspondence and shape estimation [Ira 02]. We note that the precision of shape and motion estimation relies on accuracy of the alignment of input views with respect to the reference planar surface.

The relative affine-structure framework [Sha 96] is closely related to the planar-parallax framework. The magnitude of parallax displacement of points between two views relative to a planar surface in the scene is called "the relative affine structure." The relative affine structure depends both on the "height" of a 3D point **X** from the planar surface in the scene (reference plane) and its "depth" relative to the reference camera (reference frame).

### 4.8.6 Dense Structure from Stereo

Applications such as image-based rendering of intermediate views for multi-view video require dense scene structure (depth image) reconstruction. Dense-structure estimation is not always possible with high precision using structure from motion algorithms from monocular video since presence of multiple motions by independently moving objects complicates the solution. Structure from stereo (two-views) with calibrated cameras is perhaps the most robust approach to reconstructing a dense scene structure because the epipolar geometry holds true for all pixels.

Structure from stereo (SFM) algorithms typically start with camera calibration and image rectification. Rectification refers to aligning two cameras to be co-planar, parallel to the line joining the two camera centers. This is achieved by applying a homography to both images. After image rectification, the correspondence (disparity) search is simplified to a 1D search on a horizontal line since all epipolar lines are parallel in the rectified image plane. Motion-estimation methods discussed in Section 4.5 or Section 4.6 can be used for 1D disparity estimation. A comparative evaluation of more than 20 stereo-matching algorithms is given by Scharstein and Szeliski [Sch 02].

## References

[Adi 85] Adiv, G., "Determining 3-D motion and structure from optical flow generated by several moving objects," IEEE Trans. on Patt. Anal. Mach. Intel., vol. 7, pp. 384–401, 1985.

[Agg 88] Aggarwal, J. K., and N. Nandhakumar, "On the computation of motion from sequences of images," Proc. IEEE, vol. 76, pp. 917–935, Aug. 1988.

[Ahn 03] Ahn, T.G., Y.H. Moon, and J.H. Kim, "An improved multilevel successive elimination algorithm for fast full search motion estimation," IEEE ICIP 2003.

[Alt 97] Altunbasak, Y., and A. M. Tekalp, "Closed-form connectivity-preserving solutions for motion compensation using 2-D meshes," IEEE Trans. on Image Processing, vol. 6, no. 9, pp. 1255–1269, Sep. 1997.

[Bak 04] Baker, S., and I. Matthews, "Lucas-Kanade 20 years on: A unifying framework," Int. Jour. of Computer Vision, vol. 56, no. 3, pp 221–255, Feb. 2004.

[Bak 11] Baker, S., D., Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski, "A database and evaluation methodology for optical flow," Int. Jour. of Computer Vision, vol. 92, pp. 1–31, 2011.

[Bar 94] Barron, J. L., D. J. Fleet, and S. S. Beauchemin, "Systems and experiment: Performance of optical flow techniques," Int. J. Comp. Vision, vol. 12, no. 1, pp. 43–77, 1994.

[Ber 88] Bertero, M., T. A. Poggio, and V. Torre, "Ill-posed problems in early vision," Proc. of the IEEE, vol. 76, pp. 869–889, August 1988.

[Ber 92] Bergen, J. R., P. J. Burt, R. Hingorani, and S. Peleg, "A three-frame algorithm for estimating two-component image motion," IEEE Trans. on Pattern Anal. Mach. Intel., vol. 14, no. 9, pp. 886–896, Sep. 1992.

[Bie 87] Biemond, J., L. Looijenga, D.E. Boekee, and R.H.J.M. Plompen, "A pel-recursive Wiener-based motion estimation algorithm," Signal Proc., vol. 13, pp. 399–412, 1987.

[Bie 88] Bierling, M., "Displacement estimation by hierarchical block-matching," Proc. Vis. Comm. and Image Proc., SPIE vol. 1001, pp. 942–951, 1988.

[Dai 10] Dai, Y., H. Li, and M. He, "Element-wise factorization for N-view projective reconstruction," in Proc. European Conf. on Computer Vision, 2010.

[Dub 93] Dubois, E., and J. Konrad, "Estimation of 2-D motion fields from image sequences with application to motion-compensated processing," in Motion Analysis and Image Sequence Processing, M. I. Sezan and R. L. Lagendijk, eds., Kluwer, 1993.

[Enk 88] Enkelmann, W., "Investigations of multigrid algorithms for the estimation of optical flow fields in image sequences," Comp. Vis. Graph. and Image Proc., vol. 43, pp. 150–177, 1988.

[Fau 95] Faugeras, O. D., "Stratification of three-dimensional vision: Projective, affine and metric representations," JOSA, vol. 12, no. 3, pp. 465–484, 1995.

[Fle 90] Fleet, D. J., and A. D. Jepson, "Computation of component image velocity from local phase information," Int. J. Comp. Vis., vol. 5, pp. 77–104, 1990.

[Fog 91] Fogel, S.V., "Estimation of velocity vector fields from time-varying image sequences," CVGIP: Image Understanding, vol. 53, pp. 253–287, 1991.

[For 02] Foroosh, H., J. Zerubia, and M. Berthod, "Extension of phase correlation to sub-pixel registration," IEEE Trans. Image Processing, vol. 11, no. 3, pp. 188–200, 2002.

[Gem 84] Geman, S., and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," IEEE Trans. Pattern Anal. Machine Intel., vol. 6, no. 6, pp. 721–741, 1984.

[Glo 08] Glocker, B., N. Paragios, N. Komodakis, G. Tziritas, and N. Navab, "Optical flow estimation with uncertainties through dynamic MRFs," Proc. of the IEEE Conf. CVPR, 2008.

[Har 97] Hartley, R. I., "In defense of the eight-point algorithm," IEEE Trans. on Patt. Anal. and Machine Intell., vol. 19, no. 6, pp. 580–593, Oct. 1997.

[Har 04] Hartley, R. and A. Zisserman, *Multiple View Geometry in Computer Vision, Second Edition*, New York, NY: Cambridge University Press, 2004.

[Hee 88] Heeger, D. J., "Optical flow using spatiotemporal filters," Int. J. Comp. Vis., vol. 1, pp. 279–302, 1988.

[Hil 84] Hildreth, E. C., "Computations underlying the measurement of visual motion," Artif. Intel., vol. 23, pp. 309–354, 1984.

[Hor 81] Horn, B. K. P., and B. G. Schunck, "Determining optical flow," Artif. Intell., vol. 17, pp. 185–203, 1981.

[Ira 93] Irani, M., and S. Peleg, "Motion analysis for image enhancement: Resolution, occlusion and transparency," J. Vis. Comm. and Image Rep., 4, 324–335, (1993).

[Ira 96] Irani, M., P. Anandan, J. Bergen, R. Kumar, and S. Hsu, "Efficient representations of video sequences and their applications," Sign. Proc.: Image Comm., vol. 8, pp. 327–351, 1996.

[Ira 98] Irani, M., and P. Anandan, "A unified approach to moving object detection in 2D and 3D scenes," IEEE Trans. on Patt. Anal. Machine Intel., vol. 20, no. 6, pp. 577–589, June 1998.

[Ira 02] Irani, M., P. Anandan, and M. Cohen, "Direct recovery of planar-parallax from multiple frames," IEEE Trans. on Patt. Anal. Machine Intel., vol. 24, no. 11, pp. 1528–1534, Nov. 2002.

[Jai 81] Jain, J.R., and A.K Jain, "Displacement measurement and its applications," IEEE Trans. on Communications, vol. 29, no. 12, pp. 1799–1808, Dec. 1981.

[Koe 90] Koenderink, J.J., and A.J. Van Doorn, "Affine structure from motion," Jour. of the Optical Society of America A, vol. 8, pp. 377–385, 1990.

[Kog 81] Koga, T., K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion compensated inter-frame coding for video-conferencing," Proc. Nat. Telecomm. Conf., New Orleans, LA, Nov. 1981, pp. 5.3.1–5.3.5.

[Kon 92] Konrad, J., and E. Dubois, "Bayesian estimation of motion vector fields," IEEE Trans. on Patt. Anal. and Mach. Intel, vol. 14, no. 9, pp. 910–927, Sep. 1992.

[Kug 75] Kuglin, C. D., and D. C. Hines, "The phase correlation image alignment method," Proc. of the IEEE Conf. Cybernetics and Society, New York, NY, pp. 163–165, Sept. 1975.

[Lee 90] Lee, H. C., E. J. Breneman, and C. P. Schutte, "Modeling light reflection for computer color vision," IEEE Trans. Patt. Anal. Mach. Intel., vol. 12, pp. 402–409, Apr. 1990.

[Li 95] Li, W., and E. Salari, "Successive elimination algorithm for motion estimation," IEEE Trans. on Image Processing, vol. 4, no. 1, pp. 105–107, Jan. 1995.

[Luc 81] Lucas, B. D., and T. Kanade, "An iterative image registration technique with an application to stereo vision," Int. Joint Conf. on Artificial Intell., pp. 674–679, 1981.

[Nag 86] Nagel, H. H., and W. Enkelmann, "An investigation of smoothness constraints for the estimation of displacement vector fields from image sequences," IEEE Trans. on Patt. Anal. and Mach. Intel., vol. 8, pp. 565–593, 1986.

[Nag 87] Nagel, H. H., "On the estimation of optical flow: Relations between different approaches and some new results," Artificial Intelligence, vol. 33, pp. 299–324, 1987.

[Nak 94] Nakaya, Y., and H. Harashima, "Motion compensation based on spatial transformations," IEEE Trans. Circ. and Syst. for Video Tech., vol. 4, pp. 339–356, June 1994.

[Ole 07] Oliensis, J., and R. Hartley, "Iterative extensions of the Sturm/Triggs algorithm: Convergence and nonconvergence," IEEE Trans on Pattern Anal Mach Intell., vol. 29, no. 12, pp. 2217–2233, Dec. 2007.

[Pan 13] Pan, Z., Y. Zhang, S. Kwong, X. Wang, and L. Xu, "Early termination for TZSearch in HEVC motion estimation," Proc. ICASSP 2013.

[Pen 91] Pentland, A., "Photometric motion," IEEE Trans. on Patt. Anal. Mach. Intel., vol. 13, pp. 879–890, Sep. 1991.

[Pur 12] Purnachand, N., L. N. Alves, and A. Navarro, "Fast motion estimation algorithm for HEVC," IEEE Second Int. Conf. on Consumer Electronics (ICCE), Berlin, 2012.

[Rob 83] Robbins, J. D., and A. N. Netravali, "Recursive motion compensation: A review," in Image Sequence Processing and Dynamic Scene Analysis, T. S. Huang, ed., pp. 76–103, Berlin, Germany: Springer-Verlag, 1983.

[Sch 02] Scharstein, D., and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," Int. Jour. of Computer Vision, vol. 47, no. 1–3, pp. 7–42, April 2002.

[Sha 96] Shashua, A., and N. Navab, "Relative affine structure: Canonical model for 3D from 2D geometry and applications," IEEE Trans. on Pattern Analysis and Mach. Intel., vol. 18, no. 9, pp. 873–883, Sept. 1996.

[Shu 00] Shum, H. Y., and R. Szeliski, "Construction of panoramic image mosaics with global and local alignment," Int. J. of Comp. Vision, vol. 16, no.1, pp. 63–84, 2000.

[Sny 91] Snyder, M. A., "On the mathematical foundations of smoothness constraints for the determination of optical flow and for surface reconstruction," IEEE Trans. Patt. Anal. Mach. Intel., vol. 13, pp. 1105–1114, 1991.

[Stu 96] Sturm, P., and B. Triggs, "A factorization based algorithm for multi-image projective structure and motion," Proc. of the European Conf. Comp. Vision (ECCV), pp. 709–720, April 1996.

[Sun 10] Sun, D., S. Roth, and M. J. Black, "Secrets of optical flow estimation and their principles," IEEE Conf. on Computer Vision and Pattern Recog., CVPR, June 2010.

[Sze 96] Szeliski, R., "Video mosaics for virtual environments," IEEE Computer Graphics and Applications, vol. 16, no. 2, pp. 22–30, March 1996.

[Sze 06] Szeliski, R., "Image alignment and stitching: A tutorial," Foundations and Trends in Computer Graphics and Vision, vol. 2, no. 1, pp. 1–104, 2006.

[Tan 94] Tang, Y. Y., and C. Y. Suen, "New algorithms for fixed and elastic geometric transformation models," IEEE Trans. on Image Proc., vol. 3, no. 4, pp. 355–366, July 1994.

[Ter 88] Terzopoulos, D., and K. Fleischer, "Deformable models," The Visual Computer, vol. 4, no. 6, pp. 306–331, 1988.

[Tok 96] C. Toklu, A. T. Erdem, M. I. Sezan, and A. M. Tekalp, "Tracking motion and intensity-variations using hierarchical 2-D mesh modeling for synthetic object transfiguration," Graphical Models and Image Processing, vol. 58, no. 6, pp. 553–573, Nov. 1996.

[Tom 92] Tomasi, C., and T. Kanade, "Shape and motion from image streams under orthography: A factorization method," Int. J. of Computer Vision, vol. 9, no. 2, pp. 137–154, Nov. 1992.

[Tri 99] Triggs, B., P. McLauchlan, R. Hartley, and A. Fitzgibbon, "Bundle adjustment - A modern synthesis," Proc. Int. Workshop on Vision Algorithms, Springer, pp. 298–372, 1999.

[Tsa 87] Tsai, R. Y., "A versatile camera calibration technique for 3D machine vision," IEEE Trans. on Robotics and Automation, vol. 3, no. 4, pp. 323–344, Aug. 1987.

[Ver 89] Verri, A., and T. Poggio, "Motion field and optical flow: Qualitative properties," IEEE Trans. Patt. Anal. Mach. Intel., vol. 11, pp. 490–498, May 1989.

[Wal 84] Walker, D. R., and K. R. Rao, "Improved pel-recursive motion compensation," IEEE Trans. Commun., vol. 32, pp. 1128–1134, Oct. 1984.

[Xu 08] Xu, X., and Y. He, "Improvements on fast motion estimation strategy for H.264/AVC," IEEE Trans. on Circuit and Syst. for Video Tech., vol. 18, no. 3, pp. 285–293, Mar. 2008.

[Yan 05] Yang, L., K. Yu, J. Li, and S. Li, "An effective variable block-size early termination algorithm for H.264 video coding," IEEE Trans. on Circ. and Systems for Video Tech., vol. 15, no. 6, pp. 784–788, June 2005.

[Zha 00] Zhang, Z., "A flexible new technique for camera calibration," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 22, no. 11, pp. 1330–1334, Nov. 2000.

[Zha 04] Zhang, Z., "Camera Calibration," Chapter 2 in *Emerging Topics in Computer Vision* (eds. S. B. Kang and G. Medioni), pp. 4–43, Upper Saddle River, NJ: Prentice Hall, 2004.

[Zhu 00] Zhu, S., and K. K. Ma, "A new diamond search algorithm for fast block-matching motion estimation," IEEE Trans. Image Processing, vol. 9, no. 2, pp. 287–290, Feb. 2000.

## Exercises

### Problem Set 4

4.1 Discuss the conditions under which the weak-perspective projection provides a good approximation to imaging through an ideal pinhole camera.

4.2 Homography:

**a.** Derive the homography equation (4.14) for the case of relative 3D rigid motion between a planar surface and a camera under perspective projection.

**b.** Show that Eqn. (4.14) is also valid for imaging an arbitrary rigid surface when a camera is only allowed to rotate about its $x$-, $y$-, and/or $z$-axis (no camera translation).

**4.3** Show that the model (4.18) can be obtained by linearization of the homography given by (4.14).

**4.4** What are the conditions for the existence of normal flow given by Eqn. (4.25)? Can we recover optical flow vectors from the normal flow? Discuss the relationship between the spatial-image gradients and the aperture problem.

**4.5** For a color image, the optical-flow equation (4.23) can be written for each of the R, G, and B channels separately. State the conditions on the (R, G, B) intensities so we have at least two linearly independent equations at each pixel. How valid are these conditions for general color images?

**4.6** Derive the optical-flow equation (4.23) starting from the displaced frame difference equation (4.26). Why do we need the small motion assumption for the optical-flow equation to be valid?

**4.7** Differential motion-estimation methods can estimate small displacements. Why? Explain how hierarchical-motion estimation and the iterative-refinement scheme (see Section 4.4.1) help deal with larger MVs when using the optical-flow equation for motion estimation.

**4.8** Explain how hierarchical-motion estimation helps to alleviate the aperture problem in motion estimation.

**4.9** How can we detect occlusion areas in forward- and backward-motion estimation? Discuss.

**4.10** Derive equations for direct estimation of affine-motion model parameters from image-intensity gradients without using point correspondences.

**4.11** Derive equations for direct estimation of homography from image-intensity gradients without using point correspondences. (Hint: see Appendix in [Bak 04].)

**4.12** Discuss how coordinate normalization should be implemented to obtain a stable solution for Eqn. (4.41). (Hint: See [Har 97].)

**4.13** In the phase correlation motion-estimation method:

**a.** Discuss why we observe a peak rather than an impulse when we compute Eqn. (4.48).

**b.** Discuss why the range of displacement estimates is limited to $[(-N/2)+1, N/2]$, given the DFT size is $N \times N$, for $N$ even.

## MATLAB Exercises

**4.1** Dense-motion Estimation – Optical Flow

**a.** Write a MATLAB program to compute the spatial partials $\delta s/\delta x_1$ and $\delta s/\delta x_2$ of an image

**i.** using finite differences,

**ii.** using the derivative of the Gaussian filter.

**b.** Write a MATLAB program to find the Lucas–Kanade motion vectors between two frames using $8 \times 8$ non-overlapping blocks and block-translation motion model. (Note: Do not forget to put some control statement in your program that checks the rank or the condition number of the $2 \times 2$ matrix before inversion.)

**i.** Which frame do you use to compute the spatial partials? Explain.

**ii.** Display calculated motion vectors. Are there any outlier motion vectors? Discuss.

**iii.** Warp the reference frames toward the current frame using the computed motion vectors. Display the motion-compensated frame difference. Comment on the result.

**c.** Repeat (b) using a three-level hierarchical representation of each frame and computing four iterations at each level of the hierarchy. Compare results with those in 2(b) and 2(c), and discuss the quality of the results.

**4.2** Block-Matching Motion Estimation

**a.** Design and implement the full-search motion estimation and at least one fast-motion estimation (preferably logarithmic search) method. Compare the computational load as well as the quality of both motion estimation algorithms.

**b.** Repeat using three-level hierarchical representation of each frame. Note that motion vectors from the lower resolution level multiplied by 2 in each

dimension will serve as initial estimates for the search at a higher resolution level. Discuss the advantages of hierarchical block-matching.

Your report should include a thorough discussion of your implementation and results. You should also provide a visual representation of motion vectors. The comparison of the full-search motion estimation and a fast-motion estimation method of your choice must include MAD or MSE values and a comparison of computational load (number of multiplication, addition, and compare operations).

**4.3** Parametric-Motion Estimation and Image Registration
Given two images of a distant static scene:

**a.** Affine Motion Model: Compute the spatial partials $\delta s/\delta x_1$ and $\delta s/\delta x_2$ and the temporal partial $\delta s/\delta t$ as in Problem 4.1.

**i.** Form the matrix (4.38) over the region of overlap between two images and solve Eqn. (4.39) to calculate six affine parameters to register the two images.

**ii.** Use the calculated parameters to warp the current image toward the reference image using the *imtransform* function in MATLAB to create a panaromic image.

**iii.** How do you combine pixel intensities in the overlapping region between the two frames?

**b.** Pseudo-Perspective Motion Model: Repeat part (a), this time using an eight-parameter pseudo-perspective motion model in creating an image mosaic of the scene. Compare the results of (a) and (b), and write your conclusions in the report.

Your report should include the two images stitched together (the mosaic representation) obtained by both methods and a comprehensive discussion of how you evaluate which method provides a better result.

**4.4** Homography Estimation by Feature Matching
Two images are related by a homography if they are images of a static scene captured by rotating the camera about a fixed center of projection possibly including zoom (but no camera translation) or they are images of a planar object/scene. Capture at least two images satisfying these requirements.

**a.** Select or compute a set of $N > 10$ putative-point correspondences between two images.

**b.** Normalize feature correspondences using a similarity transformation (see Section 4.5.5).

c. Estimate the homography between two images using the singular value decomposition (SVD) method.
d. Warp one image onto the other using the estimated transformation using the *maketform* and *imtransform* functions in MATLAB.
e. You can now stitch these images together to form a mosaic as in the previous exercise.

4.5 Estimation of Fundamental Matrix and Triangulation
The fundamental matrix captures the epipolar geometry between two projective views in algebraic form. Capture at least two images of a static 3D object.
a. Use a test target to obtain the camera calibration matrix $\mathbf{K}$. How do you validate the calibration parameters?
b. Select or compute a set of $N > 10$ putative point (on the object) correspondences between two images.
c. Estimate the fundamental matrix and the projective camera matrices $\mathbf{P}_1$ and $\mathbf{P}_2$ for the two views. How do you validate the camera matrices?
d. Estimate the 3D geometry of the feature points in the projective coordinates.
e. Estimate the 3D Euclidean geometry of the feature points. Your report should include the camera calibration parameters, camera matrices, and 3D plots of both the projective and Euclidean geometry of the feature points.

## MATLAB Resources

Michael J. Black, Optical Flow Software (C and MATLAB)
http://cs.brown.edu/~black/code.html

Jean-Yves Bouguet, Camera Calibration Toolbox for Matlab
http://www.vision.caltech.edu/bouguetj/calib_doc/

A. Zisserman, MATLAB Functions for Multiple View Geometry
http://www.robots.ox.ac.uk/~vgg/hzbook/code

Peter Kovesi, Model Fitting and Robust Estimation
http://www.csse.uwa.edu.au/~pk/Research/MatlabFns/index.html#robust

CHAPTER 5

# Video Segmentation and Tracking

*Video segmentation refers to partitioning images (frames) or video into spatial, temporal, or spatio-temporal regions that are homogeneous in some feature space. As with any segmentation problem, effective video segmentation requires proper feature selection and an appropriate distance measure. Temporal-segmentation partitions video into shots based on similarity of frames. Spatial-segmentation partitions each video frame into homogeneous regions, which can be achieved by segmenting each frame individually based on color similarity (intra-frame image segmentation). Spatio-temporal segmentation yields temporally connected spatial segments or object trajectories, for example, by inter-frame segmentation (based on similarity of color and motion between frames) or color and motion tracking.*

Spatial segmentation helps identify foreground and background objects, as well as color and motion boundaries and occlusion regions. Spatio-temporal segmentation aims to obtain temporally linked spatial regions (objects) over long video sequences (multiple frames). Motion or object tracking is a popular and important special case of spatio-temporal segmentation, where a specific object is segmented in space-time by causal processing, i.e., given the segmentation map at frame $k$, find the map at frame $k+1$ that is consistent with the result at frame $k$. Tracking methods may be

based on color, motion, or color and motion. Video segmentation is an integral part of many video analysis and coding tasks, including: i) advanced image/video coding and rate allocation, ii) improved motion estimation, iii) 3D motion and structure estimation with multiple objects, iv) video surveillance/understanding, v) video indexing and summarization [Dim 02], and vi) video authoring and editing.

Different features and homogeneity criteria may lead to different segmentations of the same video, e.g., color, texture, or motion segmentation [Cas 98]. Furthermore, there is no guarantee that any of the resulting automatic segmentations will be semantically meaningful, since a semantically meaningful region may have multiple colors, multiple textures, and/or multiple motions. Although semantic objects can be computed automatically in some well-constrained settings, e.g., when an object moves against a stationary background, in general semantic object segmentation requires specialized capture methods (chroma-keying) or user interaction. Specific video-segmentation methods should be considered in the context of the requirements of the application in which they are used. Factors that affect the choice of a specific segmentation method include [Cor 04]:

- Precision of segmentation: If segmentation is employed to improve the compression efficiency or rate control, then certain misalignment between segmentation results and actual object borders may not be of big concern. On the other hand, if segmentation is needed for object-based video authoring/editing or shape similarity matching, then it is of utmost importance that the estimated boundaries align with actual object boundaries perfectly, where even a single pixel error may not be acceptable.
- Complexity of content: Complexity of content can be modeled in terms of amount of camera motion, color and texture uniformity, contrast between objects, smoothness of motion, objects entering and leaving the scene, regularity of object shape along the temporal dimension, frequency of cuts and special effects, etc. Clearly, more complex scenes require more sophisticated segmentation algorithms, e.g., it is easier to detect cuts than wipes or fades.
- Real-time performance: If segmentation must be performed in real-time, e.g., for rate control in videoconferencing, then fully automatic algorithms must be used. On the other hand, one can employ semi-automatic, interactive algorithms for off-line applications such as video editing, indexing, or off-line video coding, or to obtain semantically meaningful segmentations [Izq 02].

This chapter presents several video-segmentation methods ranging from simple shot-boundary and motion-detection techniques to sophisticated

motion-segmentation, interactive object segmentation, and tracking methods. Although multi-modal signal-processing methods have been shown to be effective in some applications [Wan 00], here we only cover video modality. We start with image-segmentation methods in Section 5.1. Scene change (shot boundary) detection for temporal-video segmentation and motion-detection/background subtraction methods, where we study both frame differencing and pixel-based multi-frame background-modeling methods, are covered in Section 5.2. Motion-segmentation methods are discussed in Section 5.3. We begin with the *dominant-motion* approach, which labels independently moving regions sequentially (one at a time). We then present *multiple-motion* segmentation methods, including clustering motion parameters, maximum-likelihood segmentation, maximum *a posteriori* probability segmentation, and region-labeling methods. Simultaneous motion estimation and segmentation is discussed next, since accuracy of segmentation depends on the accuracy of the estimated motion field and vice versa. A discussion of semantically meaningful object segmentation with emphasis on chroma-keying and semi-automatic (interactive) object segmentation/tracking methods is also included. Section 5.4 provides an overview of object-tracking methods, which can be considered as spatio-temporal-object segmentation over long video sequences. Image and video matting is discussed in Section 5.5. Finally, performance evaluation of video-segmentation and tracking methods is treated in Section 5.6.

## 5.1 Image Segmentation

Segmentation groups neighboring pixels in an image or video frame together based on similarity of color, texture, and/or shape cues [Har 85, Pal 93]. We start by simple thresholding and clustering methods, which do not impose spatial-connectivity constraints on segmentation labels, in Section 5.1.1 and Section 5.1.2, respectively. The maximum *a posteriori* probability (MAP) estimation and graph-based methods, which do impose spatial-connectivity constraints, are covered in Section 5.1.3 and Section 5.1.4, respectively. Finally, we introduce active-contour models in Section 5.1.5. The reader is referred to [Sal 99, Gon 07] or other books for mathematical morphology-based methods.

### 5.1.1 Thresholding

Thresholding is a popular tool for image segmentation [Har 85]. Consider an image $s(x_1,x_2)$ composed of a light object on a dark background. Such an image has a

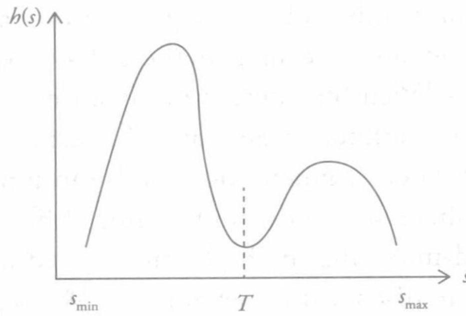



**Figure 5.1** Bi-modal histogram.

bi-modal histogram $h(s)$ as depicted in Figure 5.1. An intuitive approach to segment the object from the background, based on gray-scale information, is to select a threshold $T$ that separates these two dominant modes (peaks). The segmentation of an image into two regions is also known as binarization.

The segmentation mask or the binarized image identifies the object (denoted by 1) and background (denoted by 0) pixels:

$$z(x_1, x_2) = \begin{cases} 1 & \text{if } s(x_1, x_2) > T \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

Thresholding techniques can be divided into two broad classes: global and local. In general, threshold $T$ is a function of $T = T(x_1, x_2, s(x_1, x_2), p(x_1, x_2))$, where $p(x_1, x_2)$ is some local property of the pixel, such as the average intensity of a local neighborhood. If $T$ is selected based only on the statistics of pixel values $s(x_1, x_2)$ over the entire image, it is called a global threshold. Alternatively, $T$ can be a global threshold selected based on the statistics of both $s(x_1, x_2)$ and $p(x_1, x_2)$ to reflect certain spatial properties of the entire image. If, in addition, it depends on $(x_1, x_2)$, it is called a dynamic or adaptive threshold. Adaptive thresholds are usually computed based on a local sliding window about $(x_1, x_2)$.

### Finding the Optimum Threshold(s)

Several threshold determination methods have been proposed that are discussed in a number of review papers [Lee 90, Sez 04]. Perhaps the most popular threshold determination method for a bi-modal histogram $h(s)$ (where we have two classes of pixels in an image $s(x_1, x_2)$, e.g., foreground and background) is the Otsu method [Ots 79], which finds the optimum threshold separating two classes so that within (intra)-class variance is minimal and the between (inter)-class variance is maximal.

The within-class variance $\sigma_W^2$ as a function of the threshold $k$ is defined by a probability weighted sum of the variances of two classes $\sigma_1^2$ and $\sigma_2^2$, given by

$$\sigma_W^2(k) = P_1(k)\sigma_1^2 + P_2(k)\sigma_2^2 \tag{5.2}$$

where $P_1(k) = \sum_{i=0}^{k} p(i)$ is the probability that two classes are separated by a threshold $k$, and $p(i)$ is the value of the $i$th bin of the normalized histogram, $P_2(k) = 1 - P_1(k)$. Otsu shows that minimizing the within-class variance is equivalent to maximizing the between-class variance:

$$\sigma_B^2(k) = \sigma^2 - \sigma_W^2(k) = P_1(k)P_2(k)[\mu_1(k) - \mu_2(k)]^2 \tag{5.3}$$

where $\sigma^2$ is the combined variance, and the class means $\mu_1(k)$ and $\mu_2(k)$ are

$$\mu_1(k) = \sum_{i=0}^{k} \frac{i\,p(i)}{P_1(k)} \quad \text{and} \quad \mu_2(k) = \sum_{i=k+1}^{L-1} \frac{i\,p(i)}{P_2(k)}$$

and $L-1$ is the maximum gray level in the image. The Otsu algorithm, which performs a brute force search, can be summarized as follows.

**Otsu-Threshold Algorithm [Ots 79]**

1. Compute the normalized histogram $p(i)$, $i = 0, \ldots, L-1$ with $L$ levels.
2. Step through all possible thresholds $k = 0, \ldots, L-1$
   a. Compute $P_1(k)$, $P_2(k)$, and $\mu_1(k)$, $\mu_2(k)$
   b. Compute $\sigma_B^2(k) = P_1(k)P_2(k)[\mu_1(k) - \mu_2(k)]^2$
3. The Otsu threshold is given by $k^* = \arg\max_{0 \le k \le L-1} \sigma_B^2(k)$. If the maximum is not unique, then $k^*$ is given by the average of $k$ values corresponding to multiple maxima.

In some applications, the histogram has $K > 2$ significant modes (peaks). Then, we need $K-1$ thresholds to group pixels into $K$ segments. The extension of the Otsu method to multi-level thresholding is referred to as the multi-Otsu method [Lia 01]. Of course, reliable determination of thresholds becomes more difficult as the number of modes increases.

## 5.1.2 Clustering

In image segmentation by clustering, it is expected that feature vectors from similar-appearing image regions will form clusters in the feature space. If we consider

segmentation of an image into $K$ classes, then the segmentation label field, $z(x_1,x_2)$, assumes one of the $K$ values at each pixel, i.e., $z(x_1,x_2)=l$, $l=1,\dots,K$. In the case of scalar features, such as pixel intensities, clustering can be considered as a method of determining the $K-1$ thresholds that define the decision boundaries in the 1D feature space. With $M$-dimensional vector features, segmentation corresponds to partitioning the $M$-dimensional feature space into $K$ disjoint regions.

**$K$-Means Algorithm**

A standard procedure for clustering is to assign each sample to the class of the nearest cluster mean [Col 79, Lim 90]. In the unsupervised mode, where the cluster means are unknown, this can be achieved by an iterative procedure, known as the $K$-means algorithm. In the following, we describe the $K$-means algorithm assuming that we wish to segment an image into $K$ regions based on the gray values of the pixels. Let $\mathbf{x}=(x_1,x_2)$ denote the coordinates of a pixel and $s(\mathbf{x})$ its gray level. The $K$-means method aims to minimize the performance index:

$$J=\sum_{l=1}^{K}\sum_{\mathbf{x}\in\Lambda_l^{(i)}}\left\|s(\mathbf{x})-\mu_l^{(i+1)}\right\|^2 \tag{5.4}$$

where $\Lambda_l^{(i)}$ denotes the set of samples assigned to cluster $l$ after the $i$th iteration, and $\mu_l$ denotes the mean of the $l$th cluster. The index $J$ measures the sum of the distances of each sample from their respective cluster means. The $K$-means algorithm usually converges to a local minimum of the index $J$, hence different initializations may result in different segmentation results. The $K$-means algorithm can be summarized as follows:

1. Choose $K$ initial cluster means, $\mu_1^{(1)}, \mu_2^{(1)},\dots,\mu_K^{(1)}$, arbitrarily.
2. At the $i$th iteration assign each pixel, $\mathbf{x}$, to one of the $K$ clusters according to the relation

$$\mathbf{x}\in\Lambda_j^{(i)} \text{ if } \left\|s(\mathbf{x})-\mu_j^{(i)}\right\|<\left\|s(\mathbf{x})-\mu_l^{(i)}\right\| \text{ for all } l=1,2,\dots,K,\, l\neq j$$

   where $\Lambda_l^{(i)}$ denotes the set of samples whose cluster center is $\mu_l^{(i)}$. That is, assign each sample to the class of the nearest cluster mean.
3. Update the cluster means $\mu_l^{(i+1)}$ as the sample mean of all samples in $\Lambda_l^{(i)}$, $l=1,2,\dots,K$:

$$\mu_l^{(i+1)}=\frac{1}{N_l}\sum_{\mathbf{x}\in\Lambda_l^{(i)}} s(\mathbf{x}) \quad l=1,2,\dots,K$$

   where $N_l$ is the number of samples in $\Lambda_l^{(i)}$.

4. If $\mu_l^{(i+1)} = \mu_l^{(i)}$ for all $l=1,2,\ldots,K$, the algorithm has converged, and the procedure is terminated. Otherwise, go to step 2.

**Example.** Let $K=2$. The procedure starts with specifying two cluster means, $\mu_1$ and $\mu_2$, arbitrarily. Then, all feature points that are closer to $\mu_1$ are labeled as "$z=1$" and those closer to $\mu_2$ are labeled as "$z=2$." Next, the average of all feature points that are labeled as "1" gives the new value of $\mu_1$, and the average of those labeled as "2" gives the new value of $\mu_2$. The procedure is repeated until convergence.

The biggest challenge in *K*-means clustering is the determination of the correct number of classes, which is assumed known. In practice, the value of *K* is determined by trial and error. Different values of *K* can be tried until a desired clustering quality is achieved. To this effect, measures of clustering quality have been developed, including the within-cluster and between-cluster scatter measures [Col 79, Fig 02]. Although we have presented the *K*-means algorithm here for the case of scalar pixel features, it can be straightforwardly extended to the case of vector pixel features and/or region-based image features. The procedure provides cluster means, which can be used for other applications, such as vector quantization.

### Mean-Shift

Mean-shift (MS) is a mode-finding algorithm [Che 95]. Unlike *K*-means, the MS method does not assume prior knowledge of the number of clusters that makes it ideal for unsupervised clustering. The main idea is as follows. For each data point, $x_j$,

1. Take a window with the bandwidth parameter $h$, containing $n$ samples, centered around the data point $x_j$.
2. Compute the weighted mean (center of mass or centroid), using a kernel $g(\cdot)$ or weighting, of data within the window

$$m(x_j) = \frac{\sum_{i=1}^{n} x_i g\left(\left\|\frac{x_j - x_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{x_j - x_i}{h}\right\|^2\right)}$$

3. Shift the center of the window to the new mean and repeat the procedure until convergence.

The MS is the difference $m(x_j)-x_j$ between the weighted mean and the current center of the kernel. The stationary points of the algorithm are modes of the density function. All points associated with the same mode belong to the same cluster. Typically, MS is run at each data point or sometimes at points that are selected uniformly from the feature space [Com 02].

Combining the three steps above and assuming a Gaussian weighting, i.e., $g(x)=e^{-x}$, the mode $y_j$ for each data point $x_j$ can be computed via the following iterations:

$$y_j^{(k+1)} = \frac{\sum_{i=1}^{n} x_i e^{-\left\| \frac{y_j^{(k)} - x_i}{h} \right\|^2}}{\sum_{i=1}^{n} e^{-\left\| \frac{y_j^{(k)} - x_i}{h} \right\|^2}}, \quad k = 1, 2, \ldots \tag{5.5}$$

where $y_j^{(0)} = x_j$ is the initial center of the window (centered at the data point). It has been shown that the MS algorithm is an adaptive gradient ascent method, which is guaranteed to converge to a point (mode) where the distribution has zero gradient [Com 02]. Thus, MS steps are large in regions where data points are sparsely populated, and the steps are smaller near modes.

Although MS is a non-parametric algorithm, it requires the bandwidth parameter $h$ to be estimated. The choice of bandwidth influences the convergence rate and the number of clusters. A small $h$ can slow down convergence and may result in too many clusters. A large $h$ can speed up convergence but might merge some modes. A popular solution is to use adaptive MS where the bandwidth (size) parameter $h$ varies for each data point and is calculated using the *k-nearest neighbor* method. If $x_{i,k}$ is the $k$th-nearest neighbor of $x_i$, then the bandwidth is calculated as

$$h_i = \left\| x_i - x_{i,k} \right\| \tag{5.6}$$

A comparison of MS with $K$-means shows that $K$-means is sensitive to initialization. A wrong initialization can delay convergence or even result in wrong clusters, whereas MS is fairly robust to initialization. Likewise, $K$-means is sensitive to outliers, while MS is not. However, in higher dimensional clustering problems, the number of local maxima may be large and MS might not work well.

Clustering methods do not impose spatial-continuity constraints on the estimated segmentation labels to ensure spatial connectivity of the segments. The Bayesian segmentation, which is treated next, can be considered as clustering with statistical spatial-connectivity constraints.

### 5.1.3 Bayesian Methods

Similar to Bayesian motion estimation (Chapter 4), Bayesian segmentation methods model uncertainty in the data and prior knowledge about the desired segmentation (e.g., spatially connected segments) in terms of probability density functions (pdf), and are usually formulated as an energy minimization problem that can be solved by nonlinear optimization. This section presents the maximum *a posteriori* probability (MAP) approach, which belongs to the class of Bayesian methods. The difficulty with Bayesian methods is in specifying realistic probabilistic models and solution of the resulting nonlinear optimization problem. Numerical optimization procedures that can reach the global optimum are often time consuming (see Appendix C) and faster greedy methods can be trapped in a local minimum.

#### Maximum *A Posteriori* Probability Method

The MAP formulation takes the presence of observation noise into account explicitly by modeling observed image data as $g(\mathbf{x})=s(\mathbf{x})+v(\mathbf{x})$, where $v(\mathbf{x})$ denotes the observation noise. In vector notation, $\mathbf{g}$ denotes an $N$-dimensional vector obtained by lexicographical ordering of the monochrome image data. We wish to estimate a segmentation label field, denoted by the $N$-dimensional vector $\mathbf{z}$. A label $z(\mathbf{x}) = l$, $l = 1,2,\ldots,K$ implies that the pixel (site) $\mathbf{x}$ belongs to the $l$-th class among $K$ classes. The desired estimate of the segmentation field, $\hat{\mathbf{z}}$, is defined as the one that maximizes the *a posteriori* pdf $p(\mathbf{z}|\mathbf{g})$ of the segmentation label field, given the observed image $\mathbf{g}$. Using the Bayes rule,

$$p(\mathbf{z}\,|\,\mathbf{g}) \approx p(\mathbf{g}\,|\,\mathbf{z})\; p(\mathbf{z}) \tag{5.7}$$

where $p(\mathbf{g}|\mathbf{z})$ denotes the class-conditional pdf. The class-conditional pdf of data, given the segmentation labels $\mathbf{z}$, relates the segmentation labels to the data. The term $p(\mathbf{z})$ is the *a priori* pdf, which expresses prior expectations about the segmentation, e.g., to impose a spatial-connectivity constraint on the segmentation labels. Thus, estimation of the segmentation labels is not only dependent on the image intensity, but also constrained by the expected spatial properties imposed by the *a priori* pdf model. In the following, we first introduce the *a priori* pdf model, then proceed to examine the assumptions used in characterizing the conditional pdf.

#### A Priori *Probability Model*

Derin and Elliott [Der 87] successfully used Gibbs random field (GRF) as *a priori* probability model for segmentation labels. In order to eliminate isolated regions that

may arise in the segmentation label field, the GRF model can be designed to assign a higher probability for segmentations that have contiguous, connected regions. We review GRF representation and how to model smoothness of labels using clique potentials in Appendix B. It follows that the *a priori* pdf of the label field can be expressed by an impulse train with Gibbsian weighting:

$$p(\mathbf{z}) = \frac{1}{Q} \sum_{\omega \in \Omega} e^{-U(\mathbf{z}/T)} \delta(\mathbf{z} - \boldsymbol{\omega}) \tag{5.8}$$

where $\Omega$ is the finite sample space of the random vector $\mathbf{z}$, $\delta(\cdot)$ denotes a Dirac delta function, $T$ is the temperature parameter, the normalizing constant

$$Q = \sum_{\omega \in \Omega} e^{-U(\omega)}$$

is called the partition function, and $U(\mathbf{z})$ is the Gibbs potential defined by

$$U(\mathbf{z}) = \sum_{C \in \boldsymbol{C}} V_C(\mathbf{z})$$

Here, $\boldsymbol{C}$ is the set of all cliques, and $V_C$ is the individual clique potential function. The single-pixel clique potentials, which reflect *a priori* probabilities of different labels, can be defined as

$$V_C(z(\mathbf{x})) = \alpha_l \text{ if } z(\mathbf{x}) = l \text{ and } \mathbf{x} \in C \text{ for } l = 1, 2, \ldots, K \tag{5.9a}$$

The smaller $\alpha_l$ the higher the likelihood of region $l$, $l = 1,2 \ldots, K$. Spatial connectivity of the segmentation can be imposed by assigning two-pixel clique potentials:

$$V_C(z(\mathbf{x}_i), z(\mathbf{x}_j)) = \begin{cases} -\beta & \text{if } z(\mathbf{x}_i) = z(\mathbf{x}_j) \text{ and } \mathbf{x}_i, \mathbf{x}_j \in C \\ \beta & \text{if } z(\mathbf{x}_i) \neq z(\mathbf{x}_j) \text{ and } \mathbf{x}_i, \mathbf{x}_j \in C \end{cases} \tag{5.9b}$$

where $\beta$ is a positive constant. The larger the value $\beta$, the stronger the smoothness constraint.

### *Conditional-Probability Model*

The original image, $\mathbf{s}$, can be modeled by a mean-intensity function, denoted by the-vector $\boldsymbol{\mu}$, plus a zero-mean, white Gaussian residual process, $\mathbf{r}$, with variance $\sigma_r^2$, i.e.,

$$\mathbf{s} = \boldsymbol{\mu} + \mathbf{r} \tag{5.10a}$$

where all vectors are $N$-dimensional lexicographic ordering of the respective arrays. Then,

$$\mathbf{g} = \boldsymbol{\mu} + \boldsymbol{\xi} \tag{5.10b}$$

where $\boldsymbol{\xi} = \mathbf{r} + \mathbf{v}$ is a zero-mean Gaussian process with variance $\sigma_\xi^2 = \sigma_r^2 + \sigma_v^2$, and $\sigma_v^2$ denotes the variance of the observation noise that is also taken to be a zero-mean, white Gaussian process. We will refer to this combined term, $\boldsymbol{\xi}$, simply as the additive noise term. The original formulation by Derin and Elliott [Der 87] models the mean intensity of each image region as a constant, denoted by the scalar $\mu_l$, $l = 1, 2, \ldots, K$, for segmenting an image into $K$ regions. That is, elements of $\boldsymbol{\mu}$ attain $K$ distinct values, $\mu_l$, $l = 1, 2, \ldots, K$. Based on the model of the observed image in (5.10), the conditional-probability distribution is expressed as

$$p(\mathbf{g} \mid \mathbf{z}) \approx e^{-\sum_{\mathbf{x}} \frac{\left(g(\mathbf{x}) - \mu_{z(\mathbf{x})}\right)^2}{2\sigma_\xi^2}} \tag{5.11}$$

where $z(\mathbf{x}) = l$ designates the assignment of site $\mathbf{x}$ to region $l = 1, 2, \ldots, K$. Note that maximization of pdf (5.11) alone results in maximum likelihood (ML) image segmentation.

Substituting (5.8) and (5.11) into (5.7), the *a posteriori* density has the form

$$p(\mathbf{z} \mid \mathbf{g}) \approx e^{-\sum_{\mathbf{x}} \frac{1}{2\sigma_\xi^2}\left[g(\mathbf{x}) - \mu_{z(\mathbf{x})}\right]^2 - \sum_{C \in \mathcal{C}} V_C(\mathbf{z})} \tag{5.12}$$

We maximize the posterior density (5.12) to find estimates of $\mu_l$, $l = 1, 2, \ldots, K$, and the desired segmentation labels $\mathbf{z}$. Note that maximizing (5.12) is equivalent to minimizing the cost

$$E(\mathbf{z}) = \sum_{\mathbf{x}} \frac{1}{2\sigma_\xi^2}\left[g(\mathbf{x}) - \mu_{z(\mathbf{x})}\right]^2 + \sum_{C \in \mathcal{C}} V_C(\mathbf{z}) \tag{5.13}$$

The solution to the MAP segmentation problem can be obtained by Monte Carlo type methods. A well-known method to reach the global optimum is simulated annealing. However, because it is computationally complex, we often use a sub-optimal method, called iterated conditional mode (ICM), trading optimality for reduced computational complexity (see Appendix C).

Observe that if we turn off the spatial-smoothness constraints, i.e., neglect the second term, the result would be identical to that of the $K$-means algorithm. Thus,

we follow a procedure that is similar to that of the $K$-means algorithm, i.e., we start with an initial estimate of class means and assign each pixel to one of the $K$ classes by minimizing (5.13), then we update the class means using these estimated labels, and iterate between these two steps until convergence.

### Adaptive MAP Method

Pappas [Pap 92] has proposed an image model with slowly varying means $\mu_{z(\mathbf{x})}(\mathbf{x})$, as opposed to a constant mean $\mu_{z(\mathbf{x})}$ used by Derin and Elliott [Der 87], for modeling intensities within each image region denoted by $z(\mathbf{x})$. An adaptive clustering procedure has been proposed based on this model, where the cluster means $\mu_i$ vary slowly by pixel location $\mathbf{x}$. The MAP method can be made adaptive similarly by letting the mean of the class-conditional pdfs vary slowly. Then, the modified class-conditional pdf model becomes

$$p(\mathbf{g} \mid \mathbf{z}) \approx e^{-\sum_{\mathbf{x}} \frac{\left[g(\mathbf{x})-\mu_{z(\mathbf{x})}(\mathbf{x})\right]^2}{2\sigma_\xi^2}} \tag{5.14a}$$

where $\mu_{z(\mathbf{x})}(\mathbf{x})$ denotes the space variant mean intensity for each class $z(\mathbf{x})$ and the *a posteriori* pdf is given by

$$p(\mathbf{z} \mid \mathbf{g}) \approx e^{-\sum_{\mathbf{x}} \frac{\left[g(\mathbf{x})-\mu_{z(\mathbf{x})}(\mathbf{x})\right]^2}{2\sigma_\xi^2} - \sum_{C \in \mathcal{C}} V_C(\mathbf{z})} \tag{5.14b}$$

Again, maximizing (5.14b) is equivalent to minimizing

$$E(\mathbf{z}) = \sum_{\mathbf{x}} \frac{\left[g(\mathbf{x}) - \mu_{z(\mathbf{x})}(\mathbf{x})\right]^2}{2\sigma_\xi^2} + \sum_{C \in \mathcal{C}} V_C(\mathbf{z}) \tag{5.15a}$$

The adaptive algorithm follows a procedure that is similar to the two-step iterations described in the non-adaptive case, except that the cluster means $\mu_l(\mathbf{x})$ at site $\mathbf{x}$ for each region $l$ are estimated as the sample mean of those pixels with label $l$ within a local window about the pixel $\mathbf{x}$ (as opposed over the entire image). The following simplifications are usually performed to reduce the computational burden:

1. The space-varying mean estimates may be computed on a sparse grid and then interpolated.
2. Te optimization is performed using the ICM method (see Appendix C). Note that the ICM is equivalent to maximizing the local *a posteriori* pdf at a site $\mathbf{x}_i$, given by

$$p\left(z(\mathbf{x}_i) \mid g(\mathbf{x}_i), z(\mathbf{x}_j), \text{ all } z(\mathbf{x}_j) \in \boldsymbol{N}_{\mathbf{x}_i}\right) \approx e^{-\frac{1}{2\sigma_\xi^2}\left[g(\mathbf{x}_i) - \mu_{z(\mathbf{x}_i)}(\mathbf{x}_i)\right]^2 - \sum_{C \in \mathcal{C}} V_C(\mathbf{z})} \tag{5.15b}$$

Allowing space-varying class-means offers a couple of advantages: i) it avoids oversegmentation since fewer regions are required to segment an image into perceptually meaningful regions; ii) it avoids merging perceptually distinct regions with low local intensity contrast.

**Vector-Field Segmentation**

In most applications we deal with the segmentation of multi-channel data such as color images or motion-vector fields. The Bayesian segmentation algorithms (MAP and adaptive MAP) can be generalized to segment multi-channel data. This extension involves modeling the multi-channel data with a vector random field, where the components of the vector field (each individual channel) are assumed to be conditionally independent given the segmentation labels of the pixels. Note that we have a scalar segmentation label field, which means each vector is assigned a single label, as opposed to segmenting the channels individually.

The class-conditional probability model for the vector image field is taken as a multi-variate Gaussian distribution with a space-varying mean function. We assume that $M$ channels of multi-spectral data are available and denote them by a $P$-dimensional ($p = N \cdot M$) vector $[\mathbf{g}_1\ \mathbf{g}_2 \cdots \mathbf{g}_M]^T$, where $\mathbf{g}_j$ corresponds to the $j$th channel. A single segmentation field, $\mathbf{z}$, which is consistent with all $M$ channels of data and is in agreement with the prior knowledge, is desired. By assuming the conditional independence of the channels given the segmentation field, the conditional probability in (5.11) becomes

$$p(\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_M \mid \mathbf{z}) = p(\mathbf{g}_1 \mid \mathbf{z})\, p(\mathbf{g}_2 \mid \mathbf{z}) \ldots p(\mathbf{g}_M \mid \mathbf{z}) \tag{5.16}$$

The extension of the Bayesian methods for multi-channel data following the model (5.16) is straightforward.

## 5.1.4 Graph-Based Methods

Graph-based methods construct a graph in which the nodes represent pixels or blocks of pixels or over-segmented regions in the image and edges represent affinities (couplings) between them. The image is segmented by cutting the graph into subgraphs such that the cost, which is the sum of affinities across the cut, is minimized.

Normalizing the cost of a cut by the area of segments [Tao 07] or by a measure derived from affinities between nodes within the segments [Shi 00] avoids favoring small regions and prevents over-segmentation of the image.

Let $G = (V, E, W)$ denote a graph, where $V$ is the set of nodes, $E$ is the set of edges (links) connecting the nodes, and $W$ is an edge-affinity matrix. A pair of nodes $i$ and $j$ is connected by an edge with a weight $w(i,j) = w(j,i) > 0$, i.e., a measure of affinity (dissimilarity) between them. The graph can be partitioned into two disjoint sets, $A$ and $B = V - A$ by removing edges connecting the two parts. The degree of dissimilarity between the two sets can be computed as the total weight of removed edges, which is called a cut in graph theory:

$$cut(A,B) = \sum_{i \in A, j \in B} w(i, j) \tag{5.17}$$

There are many algorithms that solve the minimum-cut problem in polynomial time with small constants [Kol 04]. However, the minimum cut criterion favors grouping small sets of isolated nodes in the graph, because it does not contain any intragroup information, and as a result causes over-segmentation. The normalized cut (Ncut) criterion, given by

$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)} \tag{5.18}$$

where *assoc*$(A, V)$ and *assoc*$(B, V)$ denote the total connections from nodes in A and in B, respectively, to all nodes in the graph, is usually prefered since it favors equal-size regions.

In the case of the normalized cut criterion, polynomial methods, with runtime complexity $O(N^2 \log N)$, where $N$ denotes the number of nodes, exist for finding a globally optimal solution when the graph is planar [Shi 00]. When the graph is non-planar finding a globally optimal solution is NP-hard, and approximation methods are employed. The most popular solution is perhaps the one proposed by Shi and Malik [Shi 00]. Let $d_i = \sum_j w(i,j)$ denote the total affinity from node $i$ to all other nodes, $\mathbf{D}$ be an $N \times N$ diagonal matrix with diagonal entries $d_i$, $\mathbf{W}$ be an $N \times N$ matrix with entries $w(i,j)$, and $\mathbf{x}$ be an $N \times 1$ indicator vector, such that $x_i = 1$ if node $i \in A$ and $x_i = -1$ otherwise. Shi and Malik [Shi 00] have shown that

$$\min_{\mathbf{x}} Ncut(A,B) = \min_{\mathbf{y}} \frac{\mathbf{y}^T (\mathbf{D} - \mathbf{W}) \mathbf{y}}{\mathbf{y}^T \mathbf{D} \mathbf{y}} \tag{5.19}$$

with the condition $y_i \in \{1, -b\}$, where $\mathbf{y} = (\mathbf{1} + \mathbf{x}) - b(\mathbf{1} - \mathbf{x})$ and $b = \frac{\sum_{x_i>0} d_i}{\sum_{x_i<0} d_i}$. If $\mathbf{y}$ is relaxed to take on real values, then the minimization can be achieved by solving the generalized eigenvalue problem

$$(\mathbf{D} - \mathbf{W})\mathbf{y} = \lambda \, \mathbf{D}\mathbf{y} \tag{5.20}$$

where $(\mathbf{D} - \mathbf{W})$ is called the Laplacian matrix. The solution of the normalized cut problem is given by the second smallest eigenvector of this eigenvalue problem. Since the solution $\mathbf{y}$ is real-valued, a threshold must be selected to estimate the indicator vector $\mathbf{x}$. A common procedure is to conduct a search over $l$ evenly-spaced possible split thresholds to obtain the minimum $Ncut(A, B)$.

In the Shi and Malik method [Shi 00], the nodes of the graph are individual pixels. Felzenszwalb *et al.* [Fel 04] proposed an efficient implementation that runs in nearly linear time with the number of pixels in the image for superpixel formation. An alternative approach is to pre-segment the image into uniform color regions using, for example, the MS algorithm and then use the normalized cut approach with nodes of the graph taken as the initially oversegmented regions [Tao 07]. Graph-based image segmentation methods have also been extended to hierarchical image segmentation [Cou 05].

### 5.1.5 Active-Contour Models

Active-contour models (also called snakes) are parametric planar curves that snap to object/segmentation boundaries by minimizing an energy functional. The snake was first introduced by Kass *et al.* [Kas 88], who treated energy minimization as a variational calculus problem. The variational approach may suffer from the need for estimates of high-order derivatives from discrete data, unpredictable convergence of the iterative process, and inability to enforce hard constraints. Instead, Amini *et al.* [Ami 90] used discrete dynamic programming for optimization, which is numerically more stable and allows inclusion of hard constraints.

The user specifies $N$ node points, $\mathbf{x}_i = (x_{1i}, x_{2i})$, $i = 1, \ldots, N$, on the boundary of the desired object. The initial object contour is the union of contour segments that are obtained by joining these nodes with straight lines. Mathematically, the initial contour segments are given by

$$C(s) = (s - i)\mathbf{x}_{i-1} + (s - i + 1)\mathbf{x}_i, \; i - 1 \leq s \leq i \tag{5.21}$$

The initial contour is then snapped onto the desired object boundary by minimizing the total energy functional that can be expressed as the weighted sum of three energy terms

$$E = \alpha_{int} E_{int} + \alpha_{image} E_{image} + \alpha_{ext} E_{ext} \tag{5.22}$$

where $E_{int}$ represents internal forces that promote regularity (smoothness) of the curve, $E_{image}$ is the image force that pushes the contour towards high gradient edges, and $E_{ext}$ helps incorporate prior knowledge about the desired shape of the object favoring a particular shape.

We express the total energy as a sum of energy terms $E_i$ over local contour segments,

$$E_{\text{snake}} = \sum_{i=1}^{N} E_i = \alpha_{int} \sum_{i=1}^{N} E_{int,i} + \alpha_{image} \sum_{i=1}^{N} E_{image,i} + \alpha_{ext} \sum_{i=1}^{N} E_{ext,i} \tag{5.23}$$

where each contour segment links up to three nodes $\mathbf{x}_i$, $\mathbf{x}_{i-1}$, and $\mathbf{x}_{i-2}$. For closed contours, we assume that node 1 is linked to node $N$; hence, there are $N$ contour segments for $N$ nodes.

Following Amini *et al.* [Ami 90], the minimum of (5.23) is searched by a discrete multi-stage decision process (dynamic programming), where at each stage $n$ we minimize

$$E^{(n)} = \sum_{i=1}^{n} E_i, \; n = 1, \ldots, N \tag{5.24}$$

which can be performed recursively as

$$\min_j \left\{ E^{(i-1)}(j,k) + E_i(j,k,l) \right\} \tag{5.25}$$

where $j$, $k$, and $l$ refer to the indices of all possible search points (perturbations) around nodes $\mathbf{x}_{n-2}$, $\mathbf{x}_{n-1}$, and $\mathbf{x}_n$, respectively. The minimum snake energy is found at the final stage, where $E_{\text{snake}} = E^{(N)}$. Then, the optimal contour can be computed by backtracing the search points at each node that yield the minimum snake energy. In this scheme, hard constraints can be easily imposed by discarding connections that violate the constraints. The number of nodes along the contour can be increased or decreased as desired during energy optimization. If the distance between two adjacent nodes after snapping is less than a lower bound, one of them will be deleted; or if the distance exceeds an upper bound, new nodes can be added between them.

Alternative active-contour formulations based on minimization of the Mumford–Shah functional have been proposed [Cha 01, Tsa 01]. These formulations employ an active contour to model the set of discontinuities in the Mumford–Shah functional and, hence, use a global-segmentation model for stopping curve evolution as opposed to using local image gradients. Chan and Vese [Cha 01] formulate the model in terms of level-set functions and solve the associated Euler–Lagrange equations iteratively.

## 5.2 Change Detection

Change detection is employed in digital-video processing in many different contexts including temporal segmentation of video into shots, moving object detection and tracking, motion-compensated video filtering (deinterlacing, denoising), and mode selection for motion-compensated video compression (skip mode). Various change-detection methods differ according to: i) what features and scene/background models are used, ii) what distance metrics are used, and iii) what kind of threshold and scene/background-model adaptation rules are used [Rad 05]. Change-detection methods can be classified as: i) shot-boundary detection for detecting abrupt or gradual transitions between scenes, and ii) frame/background subtraction for motion or foreground object detection, which are treated in Section 5.2.1 and Section 5.2.2, respectively.

### 5.2.1 Shot-Boundary Detection

Scene-change or shot-boundary detection is a temporal segmentation. Temporal discontinuities may be abrupt (cuts) or gradual (special effects, such as wipes and fades). It is easier to detect cuts than special effects. Shot-boundary detection methods locate global temporal discontinuities, i.e., frames across which large differences are observed in some feature space, usually a combination of color and motion [Jia 98, Gar 00, Kop 01, Lie 01, Han 02].

#### Pixel-Difference Methods

The simplest approach for detecting temporal discontinuities is to quantify frame differences in the pixel-intensity domain. If a pre-determined number of pixels exhibit differences larger than a threshold value, then a "cut" can be declared. Clearly, this method is sensitive to the presence of camera motion, noise, and compression artifacts in the video. A more robust approach may be to divide each frame into rectangular blocks, compute statistics of each block such as the mean and variance

independently, and then check the count of blocks with changing statistics against a set threshold. Applying low-pass filtering to each frame prior to computing frame differences or block statistics can improve robustness.

### Histogram-Based Methods

Histogram differences are generally more robust than pixel-wise or block-wise intensity differences. We compute $n$-bin color histogram, $h_k(i)$, i = 1,..., $n$, for each frame $k$. Various measures and tests have been developed to quantify similarity or dissimilarity of histograms. These include the histogram-intersection measure, chi-square test, and Kolmogorov–Smirnov test [Kop 01]. A closely related approach is to detect changes in the counts of edge pixels in successive frames, i.e., similarity of edge histograms. Although they are effective at detecting cuts and fades, neither histogram differences nor intensity differences can usually differentiate between wipes and camera motion, such pans and zooms. Detection of these special effects requires a combination of histogram difference and camera-motion estimation. Global motion can be estimated and frames are motion compensated before computation of the features [Bou 99]. Another approach to detect gradual changes is the so-called twin-comparison method [Zha 93], which can be used with different features. A lower threshold is used to detect abrupt scene changes, while a higher threshold is used to detect the actual position of gradual ones.

There also exist shot-boundary detection algorithms for specific domains, such as surveillance video [Str 00], sports video [Eki 03], and movies [Ham 95, Sun 02]. Sports video is arguably one of the most challenging domains for robust shot-boundary detection due to: i) the existence of a strong color correlation between successive shots, since a single dominant color background, such as the soccer field, may be seen in successive shots; ii) existence of large camera and object motions; iii) existence of many gradual transitions, such as wipes and dissolves. Ekin *et al.* [Eki 03] observed that gradual transitions in sports video are not accurately detected by simple algorithms using a single feature and proposed using two features, the absolute difference between two frames of the ratio of dominant colored pixels to total number of pixels, and color histogram dissimilarity, measured by histogram intersection, for reliable shot-boundary detection.

### Compressed-Domain Methods

Videos are stored and transmitted in compressed form. Detection of scene changes in real-time may pose a challenge in some applications since decompressing and processing video data sequentially requires significant computational resources,

motivating scene segmentation in the compressed domain (without complete bit-stream decoding) [Lel 03]. DC images that are spatially reduced versions of original video frames can be constructed from the DC coefficient of each $8 \times 8$ block in intra-coded pictures [Yeo 95]. Successful results have been obtained for detection of both abrupt and gradual scene changes using only DC images [Sal 99].

## 5.2.2 Background Subtraction

Motion-detection or background-subtraction methods segment each frame into changed and unchanged regions subject to different requirements in varying contexts. In motion-compensated filtering and compression, motion detection needs to detect whether the value of the current pixel is significantly different from the value of the co-located pixel in the previous/reference frame/field. In the context of computer vision or scene analysis, background subtraction needs to segment a scene into meaningful foreground and background regions in order to detect and track moving objects.

This section treats the background-subtraction problem, starting with a discussion of frame-differencing methods. Motion detection can be considered as a special case of frame differencing, where the background model is set equal to the previous (or a reference) frame/field. We then introduce adaptive background modeling using a mixture of Gaussians, which is more robust to variations in the background [Sta 99]. Finally, we present a method, visual background extractor ViBe [Bar 11], that integrates several concepts.

### Frame Differencing

The simplest method to detect changes between two properly registered frames is to analyze the frame difference (FD) image [Chi 02], which shows pixel-by-pixel differences between the current frame $s_k(\mathbf{x})$ and a model frame $M_k(\mathbf{x})$, given by

$$FD_k(\mathbf{x}) = s_k(\mathbf{x}) - M_k(\mathbf{x}) \tag{5.26}$$

where $\mathbf{x} = (x_1, x_2)$ is pixel location. Assuming a static camera and that illumination variations are accounted for in $M_k(\mathbf{x})$, we can distinguish non-zero differences that are due to noise from those that are due to genuine motion by thresholding the FD as

$$z_k(\mathbf{x}) = \begin{cases} 1 & \text{if } |FD_k(\mathbf{x})| > T \\ 0 & \text{otherwise} \end{cases} \tag{5.27}$$

where $T$ is a fixed threshold. Here, $z_k(\mathbf{x})$ is called segmentation label field, which is equal to "1" for changed regions and "0" otherwise. The value of the threshold $T$ can be determined by an optimal threshold-determination algorithm or as a function of the variance of noise. This pixel-wise thresholding is generally followed by post-processing to eliminate the isolated labels. Post-processing includes forming 4- or 8-connected regions and discarding labels with less than a predetermined number of connections and morphological filtering of changed and unchanged region masks.

The model frame $M_k(\mathbf{x})$ is chosen according to the requirements of the problem as:

- *Motion Detection by Successive Frame Differences:* For simple motion/change detection, the model frame/field is set equal to the immediate-previous or a past reference frame/field. Although successive differences yield satisfactory results for motion-adaptive filtering and mode selection in video compression, it often does not produce a consistent moving object mask for object tracking, since the model frame contains both moving objects and the background. For example, the uncovered background belongs to the changed region and, hence, appears as part of the moving objects.
- *Background Subtraction Using a Fixed-Reference Frame:* For moving-object detection and tracking, if a controlled reference frame that consists of only a background (without any foreground object) is available, it can be used as a fixed-model frame for all $k$. For example, if we are interested in monitoring a hallway using a fixed camera, an image of the hallway when it is empty may be used as a fixed-reference frame. However, this choice is not robust against changes in scene illumination and temporal background clutter, which is often present in outdoor scenes.
- *Background Subtraction Using a Filtered-Model Frame:* As a compromise between these two choices, a model frame can be reconstructed by filtering, e.g., mean or median filtering of last $N$ frames, where typically $N = 10$. The mean filter can be implemented as a weighted running average of past frames given by [Ira 94]

$$\mathrm{M}_k(\mathbf{x}) = \begin{cases} (1-\alpha)s_k(\mathbf{x}) + \alpha\ \mathrm{M}_{k-1}(\mathbf{x}) & k = 1, \ldots \\ s_0(\mathbf{x}) & k = 0 \end{cases}$$

where $0 < \alpha < 1$ is the learning coefficient. After processing a few frames, the unchanged regions in $\mathrm{M}_k(\mathbf{x})$ maintain their sharpness with a reduced level of noise, while changed regions are blurred and average out to the background value. The temporal integration increases the likelihood of eliminating spurious labels, thus resulting in spatially contiguous regions.

In practice, a simple FD analysis is not satisfactory for background subtraction for two reasons: First, a uniform intensity region may be interpreted as stationary even if it is part of a moving object (the aperture problem). It may be possible to avoid the aperture problem using multi-resolution analysis, since uniform intensity regions are smaller at lower resolution levels. Second, the intensity difference due to motion is affected by the magnitude of the spatial gradient in the direction of motion. This can be addressed by considering a locally normalized FD function [Ira 94] or locally adaptive thresholding [Ner 98]. An improved multi-resolution frame difference analysis that addresses both concerns can be summarized as:

1. Construct a Gaussian pyramid where each frame is represented in multiple resolutions. Start processing at the lowest resolution level.
2. For each pixel at the present resolution level, compute the normalized frame difference given by [Ira 94]

$$FDN_k(\mathbf{x}) = \frac{\sum_{\mathbf{x}\in N} |s_k(\mathbf{x}) - M_k(\mathbf{x})| \, |\nabla M_k(\mathbf{x})|}{\sum_{\mathbf{x}\in N} |\nabla M_k(\mathbf{x})|^2 + c} \tag{5.28}$$

   where $\boldsymbol{N}$ denotes a local neighborhood of the pixel $\mathbf{x}$, $\nabla M_k(\mathbf{x})$ denotes the gradient of image intensity at pixel $\mathbf{x}$, and $c$ is a constant to avoid numerical instability. If the normalized difference is high (indicating that the pixel is moving), replace the normalized difference from the previous resolution level at that pixel with the new value. Otherwise, retain the value from the previous resolution level.
3. Repeat step 2 for all resolution levels.
4. Finally, apply thresholding to the normalized motion-detection function at the highest resolution level.

Temporal memory can be incorporated into the decision process by considering accumulative differences over a sequence of $N$ frames. An accumulative difference value for each pixel is incremented by one if the difference between the current frame and the model frame at that pixel location is bigger than a threshold. Thus, pixels with higher counter values are more likely to correspond to changed regions.

### Adaptive Background Modeling by Mixture of Gaussians

A highly popular approach to modeling more complex backgrounds has been to consider each pixel as a temporal-pixel process, such that the distribution of the pixel intensity is modeled by a mixture of $K$ Gaussians, where $K$ varies between

3 and 5 [Sta 99, Kae 02]. The model is based on the assumption that each observable pixel value is generated by one of $K$ different hidden states, representing different background or foreground objects/surfaces visible at that pixel, where the pixel-intensity distribution given the state $k$ (due to temporal texture or noise and small illumination changes) is a Gaussian with mean $\mu_k$ and variance $\sigma_k^2$. The parameters $\omega_k$, $k = 1, \ldots, K$, indicate the *a priori* probability that the pixel is generated by state $k$, where $\sum_{k=1}^{K} \omega_k = 1$.

The mixture model enables learning repetitive pixel variations by maintaining $K$ model distributions for each pixel; hence, a background model is maintained even if it is temporarily replaced by another distribution. Note that a single Gaussian for each pixel (i.e., $K = 1$) would be sufficient to model the observation noise, under the unrealistic assumption that each pixel resulted from a single surface under particular lighting.

The number of states $K$, the mean $\mu_k$, and the variance $\sigma_k^2$ of each Gaussian, as well as the *a priori* probability $\omega_k$ of each Gaussian, are all unknowns and must be estimated from the observed pixel data. Given $K$, the maximum likelihood solution to this problem can be obtained by the expectation-maximization (EM) algorithm [Dem 77]. The EM algorithm works by iterating between two steps: E-step: find the expected value of the hidden state using the observed data and current estimates of the parameters; and M-step: calculate the maximum likelihood estimates of the parameters using the observed data and current estimate of the hidden state.

Stauffer–Grimson [Sta 99] provides an online approximation to the EM solution that can deal with lighting changes, repetitive variations, tracking through cluttered regions, and introduction or removal of objects from the scene. The method has two input parameters: the learning coefficient $\alpha$ and the proportion $T$ of the data that should belong to the background. Let the value of a pixel at frame $t$ be denoted by $s_t(\mathbf{x})$. At each frame $t$ and at each pixel $\mathbf{x}$, we determine the state $k$ that generates pixel $\mathbf{x}$ such that $s_t(\mathbf{x})$ is generated by state $k$ if it is within $2.5\sigma_k$ of the mean $\mu_k$. Then, the mean and variance of the $k$th Gaussian are updated as

$$\mu_{k,t} = (1-\rho)\mu_{k,t-1} + \rho\, s_t(\mathbf{x})$$

$$\sigma_{k,t}^2 = (1-\rho)\sigma_{k,t-1}^2 + \rho\, \left(s_t(\mathbf{x}) - \mu_{k,t}\right)^2$$

$$\rho = \alpha\, \mathcal{N}(s_t(\mathbf{x}) \mid \mu_{k,t}, \sigma_{k,t}^2)$$

$$\omega_{k,t} = (1-\alpha)\omega_{k,t-1} + \alpha\, M_{k,t}$$

where $\mathcal{N}(s_t(\mathbf{x}) \mid \mu_{k,t}, \sigma^2_{k,t})$ denotes a Gaussian with mean $\mu_{k,t}$ and variance $\sigma^2_{k,t}$, and $M_{k,t} = 1$ for the matching Gaussian and zero for all others. If the value of $s_t(\mathbf{x})$ does not match any of the existing Gaussians, the least probable distribution is replaced by a new one with the mean $s_t(\mathbf{x})$, a high variance, and low probability $\omega_{k,t}$.

The Gaussians with the most supporting evidence (highest *a priori* probability) and least variance belong to the background model. Hence, the Gaussians are ordered from the highest to lowest value of $\omega_{k,t}/\sigma_{k,t}$, and the first $B$ are chosen as the background model, where

$$B = \arg\min \sum_{i=1}^{b} \omega_i > T$$

Background modeling by adaptive mixture of Gaussians offers several advantages including: i) a different decision threshold applies to each pixel, ii) thresholds vary by time, and iii) multiple background models can co-exist. Perhaps the most important drawback of the method is that it requires an initialization phase, and learning can be slow especially in the presence of slow-moving objects. Improvements for faster learning and shadow modeling are proposed [Kae 02].

The case of a moving camera can be handled similarly, once the global camera motion between successive frames is estimated and compensated for [Mec 98].

### Spatial and Temporal Consistency

Another consideration is to enforce consistency of the boundaries of the changed regions with spatial-edge locations at each frame. This may be accomplished by first segmenting each frame into uniform color and/or texture regions. Next, each region resulting from the spatial segmentation is labeled as changed or unchanged as a whole as opposed to labeling each pixel independently. Region-labeling decisions may be based on the number of changed and unchanged pixels within each region or thresholding the average value of the frame differences within each region.

The boundary of changed regions is smoothed by a relaxation method using local adaptive thresholds [Aac 93]. Memory is incorporated by re-labeling unchanged pixels that correspond to changed locations in one of the last $L$ frames to ensure temporal continuity of changed regions across frames. The depth of the memory $L$ may be adapted to scene content to limit error propagation. Finally, post-processing to obtain the final changed and unchanged masks eliminates small regions.

### ViBe Algorithm [Bar 11]

The visual backgound extractor (ViBe) is a universal motion-detection or background extraction method that incorporates a pixel-based temporal model and a

policy for spatial propagation of background pixel values. In the following, we present the background model as well as model initialization and model update policies of the ViBE algorithm.

### *Background Model*

Let $s(\mathbf{x})$ denote the value of pixel $\mathbf{x}$ and $s_i$ denote a background sample value with index $i$. Each background pixel $\mathbf{x}$ is modeled by a collection of $N$ background samples $M(\mathbf{x}) = \{s_1, s_2, \dots, s_N\}$ taken from previous frames. To classify a pixel value $s(\mathbf{x})$ according to its corresponding model $M(\mathbf{x})$, we compare it to the closest values within the set of samples by defining a sphere of radius $R$ centered on $s(\mathbf{x})$. The pixel value $s(\mathbf{x})$ is then classified as background if the cardinality of the set intersection of this sphere and the collection of model samples $M(\mathbf{x})$ is larger than or equal to a given threshold. The classification of a pixel value $s(\mathbf{x})$ involves the computation of $N$ distances between $s(\mathbf{x})$ and model samples, and of $N$ comparisons with a thresholded Euclidean distance $R$. The ViBe model is determined by two parameters only: the radius $R$ of the sphere and the minimal cardinality. A radius $R = 20$ (for monochromatic images) and a cardinality of 2 have been suggested as universal parameters, with no need to adapt them from frame-to-frame or from pixel-to-pixel. The classification step of ViBe compares the current pixel value $s_t(\mathbf{x})$ to the samples in the background model of the previous frame, $M_{t-1}(\mathbf{x})$.

### *Background-Model Initialization*

Many techniques in the literature need several dozen frames to initialize their models. In order to respond to sudden illumination or scene changes, it is desirable to be able to initialize the background model from a single frame, so the existing background model can be discarded and a new model initialized instantaneously. To this effect, ViBe assumes that neighboring pixels share a similar temporal distribution and populates $M(\mathbf{x})$ with values in the spatial neighborhood of each pixel. The size of the neighborhood needs to be large enough to have a sufficient number of different samples, while the statistical correlation between pixel values decreases as the size of the neighborhood increases. The only drawback is that the presence of a moving object in the first frame will introduce an artifact called a ghost (i.e., a set of connected points, detected as in motion, but not corresponding to any real moving object). In this particular case, the ghost is caused by the undesired initialization of pixel models with samples coming from the moving object. We note that the ghost fades over time through the regular model update process, which learns the real background.

### *Background-Model Update*

The model update step decides which samples will be memorized by the model and for how long. The classical model update policy is to discard and replace the oldest value after a number of frames. The ViBe update method incorporates three novel policies so the model can adapt to changing conditions:

1. ViBe chooses the sample to be discarded randomly according to a uniform pdf, instead of always removing the oldest sample from the pixel model $M(\mathbf{x})$.
2. Random time sub-sampling: The random replacement policy allows the pixel model to cover a large (theoretically infinite) time window with a limited number of samples, but in the presence of periodic or pseudo-periodic background motions, the use of fixed sub-sampling intervals might prevent the background model from properly adapting to these motions. So when a pixel value has been classified as belonging to the background, a random process determines whether this value is used to update the corresponding pixel model.
3. A mechanism that propagates background pixel samples spatially to ensure spatial consistency and to allow the adaptation of the background pixel models that are masked by the foreground: ViBe considers that neighboring background pixels share a similar temporal distribution and that a new background sample of a pixel should also update the models of neighboring pixels. According to this policy, background models hidden by the foreground will be updated with background samples from neighboring pixel locations from time to time. This allows a spatial diffusion of information regarding the background evolution that relies on samples classified exclusively as background. ViBe's background model is thus able to adapt to a changing illumination and to structural evolutions (added or removed background objects), while relying on a strict conservative update scheme.

### Other Approaches

The dominant-motion segmentation approach, as discussed in Section 5.3.1, can also be used for foreground/background separation, assuming that dominant motion originates either from the background or a foreground object in the scene. Irani and Anandan [Ira 98] propose using planar parallax to detect moving objects in 2D/3D scenes, e.g., when a scene is approximately flat or when the camera undergoes only rotation and zoom.

## 5.3 Motion Segmentation

Motion-segmentation (also known as optical-flow segmentation) methods label pixels (or optical-flow vectors) at each frame that are associated with independently moving parts of a scene. The region boundaries may or may not be pixel-accurate or semantically meaningful. For example, a single object with articulated motion may be segmented into multiple regions. Occlusion and aperture problems are mainly responsible for misalignment of motion and actual object boundaries. Furthermore, model misfit possibly due to deviation of the surface structure from a plane generally leads to over-segmentation of the motion field. While it is possible to achieve fully automatic-motion segmentation with limited accuracy for certain content domains, semantically meaningful object segmentation generally requires user interaction to define the object of interest in at least some key frames as discussed in Section 5.5.

Motion segmentation is closely related to two other problems, motion (change) detection and motion estimation. Change detection, discussed in Section 5.2, is a special case of motion segmentation with only two regions, changed and unchanged regions (in the case of a static camera) or global and local motion regions (in the case of a moving camera). Change detection in the case of a moving camera and general motion segmentation requires some sort of global and/or local motion estimation, either explicitly or implicitly. Motion detection and segmentation are also plagued with the same two fundamental limitations associated with motion estimation: occlusion and aperture problems (see Chapter 4). For example, pixels in a flat-image region may appear stationary even if they are moving due to the aperture problem (hence, the need for hierarchical methods), and/or erroneous labels may be assigned to pixels in covered or uncovered image regions due to the occlusion problem.

In general, application of standard image segmentation methods directly to estimated optical-flow vectors may not yield meaningful results, since an object moving in 3D usually generates a spatially varying optical-flow field [Adi 85]. For example, in a rotating object, there is no flow at the center of the rotation, and the magnitude of flow vectors grows as we move away from the center of rotation. Therefore, a parametric model-based approach, where we assume that the motion field can be described by a set of $K$ parametric models, is usually adopted. In parametric-motion segmentation, the model parameters are the motion features. Then, motion-segmentation algorithms aim to determine the number of motion models that can adequately describe a scene, type/complexity of these motion models, and the spatial support of each motion model. The most commonly used types of parametric models are affine, perspective, and quadratic mappings, which assume a 3D planar

surface in motion. In the case of a non-planar object, the resulting optical flow can be modeled by a piecewise affine, perspective, or quadratic-flow field if we approximate the object surface by a union of a small number of planar patches. Since each independently moving object and/or planar patch will best fit a different parametric model, the parametric approach may lead to over-segmentation of motion in the case of non-planar objects.

## 5.3.1 Dominant-Motion Segmentation

Segmentation by dominant motion refers to extracting one object (with the dominant motion) from the scene at a time [Bur 91, Ber 91, Wu 93, Hsu 94, Ira 94, Aye 95]. Dominant-motion segmentation can be considered as a hierarchically structured top-down approach that starts by fitting a single parametric-motion model to the entire frame and then partitions the frame into two regions, those pixels that are well represented by this dominant-motion model and those that are not. The process converges to the dominant-motion model in a few iterations, each time fitting a new model to only those pixels that are well represented by the motion model in the previous iteration. The dominant motion may correspond to the camera (background) motion or a foreground object motion, whichever occupies a larger area in the frame. The dominant-motion approach may also handle separation of individually moving objects. Once the first dominant object is segmented, it is excluded from the region of analysis, and the entire process is repeated to define the next dominant object. This is unlike the multiple-motion segmentation approaches discussed in the next section, which start with an initial segmentation mask (usually with many small regions) and refine them according to some criterion function to form the final mask. It is worth noting that the dominant-motion approach is a direct method that is based on spatio-temporal-image intensity gradient information. This is in contrast to first estimating the optical-flow field between two frames and then segmenting the image based on the estimated optical-flow field.

### Segmentation Using Two Frames

Motion estimation in the presence of more than one moving object with unknown supports is a difficult problem. Burt *et al.* [Bur 91] showed that the motion of 2D translating objects can be accurately estimated by using a multi-resolution iterative approach, even in the presence of other independently moving objects without prior knowledge of their supports. This is, however, not always possible with more sophisticated motion models (e.g., affine and perspective), which are more sensitive to presence of other moving objects in the region of analysis.

Irani *et al.* [Ira 94] proposed multi-stage parametric modeling of the dominant motion. In this approach, first a translational-motion model is employed over the whole image to obtain a rough estimate of the support of the dominant motion. The complexity of the model is then gradually increased to affine and projective models with refinement of the support of the object in between. The parameters of each model are estimated only over the support of the object, based on the previously used model. The procedure can be summarized as follows:

1. Compute the dominant 2D translation vector $\mathbf{d} = (d_1, d_2)$ over the whole frame by solving

$$\begin{bmatrix} I_{x_1}^2 & I_{x_1} I_{x_2} \\ I_{x_1} I_{x_2} & I_{x_2}^2 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \end{bmatrix} = \begin{bmatrix} -I_{x_1} I_t \\ -I_{x_2} I_t \end{bmatrix} \tag{5.30}$$

   where $I_{x_1}$, $I_{x_2}$, and $I_t$ denote partials of image intensity with respect to $x_1$, $x_2$, and $t$. In case the dominant motion is not a translation, the estimated translation becomes a first-order approximation of the dominant motion.
2. Label all pixels that correspond to the estimated dominant motion as follows:
   a. Register the two images using the estimated dominant-motion model. The dominant object appears stationary between the registered images, while other parts of the image do not.
   b. Detect and label stationary regions between the registered images, which can be done by the multi-resolution change-detection algorithm discussed in Section 5.2.
   c. In addition to the normalized frame difference (5.28), we define a motion-reliability measure as the reciprocal of the condition number of the coefficient matrix in (5.30), given by [Ira 94]

$$R(\mathbf{x}, k) = \frac{\lambda_{\min}}{\lambda_{\max}} \tag{5.31}$$

   where $\lambda_{\min}$ and $\lambda_{\max}$ are the smallest and largest eigenvalue of the coefficient matrix. A pixel is classified as stationary at a resolution level if its normalized frame difference is low and its motion reliability is high. This step defines the new region of analysis for the next step.
3. Estimate the parameters of a higher-order motion model (affine, perspective, or quadratic) over the new region of analysis as in [Ira 94].
4. Iterate over steps 2 and 3 until a satisfactory segmentation is attained.

### Temporal Consistency

Temporal consistency of estimated dominant object regions can be facilitated by defining an internal representation image [Ira 94]:

$$\bar{s}(\mathbf{x},k) = \begin{cases} (1-\alpha)\ s(\mathbf{x},k) + \alpha\ \text{warp}(\bar{s}(\mathbf{x},k-1), s(\mathbf{x},k)), & k = 1,\ldots \\ s(\mathbf{x},0) & k = 0 \end{cases} \quad (5.32)$$

where warp($A$, $B$) denotes warping image A toward image B according to the dominant-motion parameters estimated between images A and B, and $0 < \alpha < 1$. As in the case of background subtraction, the unchanged regions in $\bar{s}(\mathbf{x}, k)$ maintain their sharpness with a reduced level of noise, while the changed regions are blurred after processing a few frames.

The algorithm to track the dominant object across multiple frames can be summarized as follows [Ira 94]. For each frame:

1. Compute the dominant-motion parameters between the internal representation image $\bar{s}(\mathbf{x},k)$ and the new frame $s(\mathbf{x},k)$ within the support $\mathbf{M}_{k-1}$ of the dominant object at the previous frame.
2. Warp the internal representation image at frame $k-1$ toward the new frame according to the computed motion parameters.
3. Detect the stationary regions between the registered images as described in Section 5.2.2 using $\mathbf{M}_{k-1}$ as an initial estimate to compute the new mask $\mathbf{M}_k$.
4. Update the internal representation image using Eqn. (5.32).

Comparing each new frame with the internal representation image as opposed to the previous frame allows the method to track the same object. This is because the noise is significantly filtered in the internal representation image of the tracked object, and the image gradients outside the tracked object are lowered due to blurring. Note that there is no temporal motion-constancy assumption in this tracking scheme.

### Multiple Motions

Multiple-object segmentation can be achieved by repeating the same segmentation procedure on the residual image after each dominant object is extracted. Once the first dominant object is segmented and tracked, the procedure can be repeated recursively to segment and track the next dominant object after excluding all pixels belonging to the first object from the region of analysis. Hence, the method is capable of segmenting multiple moving objects in a top-down fashion if a dominant motion exists at each stage.

Some difficulties with the dominant-motion approach have been reported when there's no overwhelmingly dominant motion. In the absence of competing motion models, the dominant-motion approach can lead to arbitrary decisions (relying upon absolute threshold values), especially when the motion measure indicates unreliable motion vectors (in low spatial-gradient regions). Sawhney *et al.* [Saw 95] proposed robust estimators to partially alleviate this problem.

## 5.3.2 Multiple-Motion Segmentation

Multiple-motion segmentation methods allow multiple-motion models to compete against each other at each decision site. They consist of three basic steps, which are strongly interrelated: estimation of the number $K$ of independent motions, estimation of model parameters for each motion, and determination of support of each model (segmentation labels). If we assume we know the number $K$ of motions and the $K$ sets of motion parameters, then we can determine the support of each model. The segmentation procedure then assigns the label of the parametric-motion vector that is closest to the estimated flow vector at each site. Alternatively, if we assume we know the value of $K$ and a segmentation map consisting of $K$ regions, the parameters for each model can be computed in the least-squares sense (either from estimated flow vectors or from spatio-temporal intensity values) over the support of the respective region. But since both the parameters and supports are unknown in reality, we have a chicken-egg problem; i.e., we need to know the motion-model parameters to find the segmentation labels, and the segmentation labels are needed to find the motion-model parameters.

Various approaches exist in the literature for solving this problem by iterative procedures. They may be grouped as: segmentation by clustering in the motion-parameter space [Adi 85, Wan 94, Kru 96], maximum-likelihood (ML) segmentation [Aye 95, Wei 96, Alt 98], and maximum *a posteriori* probability (MAP) segmentation [Mur 87], which are discussed next.

### Clustering in the Motion-Parameter Space

A simple segmentation strategy is to first determine the number $K$ of models (motion hypotheses) that are likely to be observed in a sequence and then perform clustering in the model parameter space (e.g., a six-dimensional space for the case of affine models) to find $K$ models representing the motion. In the following, we study two distinct approaches in this class: the $K$-means method and the Hough transform method.

### K-*Means Method*

Wang and Adelson (W-A) [Wan 94] employed $K$-means clustering for segmentation in their layered video representation. The W-A method starts by partitioning the image into non-overlapping blocks uniformly distributed over the image, and fits an affine model to the estimated motion field (optical flow) within each block. In order to determine the reliability of the parameter estimates at each block, the sum of squared distances between the synthesized and estimated flow vectors is computed as

$$\varepsilon^2 = \sum\nolimits_{\mathbf{x} \in B} \|\mathbf{v}(\mathbf{x}) - \tilde{\mathbf{v}}(\mathbf{x})\|^2 \tag{5.33}$$

where $\boldsymbol{B}$ refers to a block of pixels. If the flow within the block complies with a single affine model, the residual will be small. On the other hand, if the block falls on the boundary between two distinct motions, the residual will be large. The motion parameters for blocks with acceptably small residuals are selected as seed models. Then, the seed model parameter vectors are clustered to find the $K$ representative affine-motion models. The clustering procedure can be described as: Given $N$ seed affine parameter vectors $\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_N$, where

$$\mathbf{A}_n = \begin{bmatrix} a_{n,1} \\ a_{n,2} \\ a_{n,3} \\ a_{n,4} \\ a_{n,5} \\ a_{n,6} \end{bmatrix} \quad n = 1, \ldots, N \tag{5.34}$$

find $K$ cluster centers $\bar{\mathbf{A}}_1, \bar{\mathbf{A}}_2, \ldots, \bar{\mathbf{A}}_K$, where $K \ll N$, and the label $k$, $k = 1, \ldots, K$, assigned to each affine parameter vector $\mathbf{A}_n$, which minimizes

$$\sum_{n=1}^{N} D\left(\mathbf{A}_n, \bar{\mathbf{A}}_k\right)$$

The distance measure $D$ between two affine parameter vectors $\mathbf{A}_n$ and $\bar{\mathbf{A}}_k$ is given by

$$D(\mathbf{A}_n, \bar{\mathbf{A}}_k) = \mathbf{A}_n^T \; \mathbf{M} \; \bar{\mathbf{A}}_k \tag{5.35}$$

where $\mathbf{M}$ is a $6 \times 6$ scaling matrix.

The solution to this problem is given by the well-known $K$-means algorithm, which consists of the following iteration:

1. Initialize $\bar{\mathbf{A}}_1, \bar{\mathbf{A}}_2, \ldots, \bar{\mathbf{A}}_K$ arbitrarily.
2. For each seed block $n$, $n = \{1,2,\ldots,N\}$, find $k$ given by

$$k = \arg\min_s D\left(\mathbf{A}_n, \bar{\mathbf{A}}_s\right)$$

   where $s$ takes values from the set $\{1,2, \ldots, K\}$. It should be noted that if the minimum distance exceeds a threshold, then the site is not labeled, and the corresponding flow vector is ignored in the parameter update that follows.
3. Define $\mathbf{S}_k$ as the set of seed blocks whose affine parameter vector is closest to $\bar{\mathbf{A}}_k$, $k = 1, \ldots, K$. Then, update the class means

$$\bar{\mathbf{A}}_k = \frac{\sum_{n \in \mathbf{S}_k} \mathbf{A}_n}{\sum_{n \in \mathbf{S}_k} 1}$$

4. Repeat steps 2 and 3 until the class means $\bar{\mathbf{A}}_k$ do not change by more than a pre-defined amount between successive iterations.

Statistical tests can be applied to eliminate parameter vectors that are considered as outliers.

Once $K$ cluster centers are determined, a label-assignment procedure is employed to assign a segmentation label $z(\mathbf{x})$ to each pixel $\mathbf{x}$ as

$$z(\mathbf{x}) = \arg\min_k \left\| \mathbf{v}(\mathbf{x}) - \boldsymbol{P}(\bar{\mathbf{A}}_k; \mathbf{x}) \right\|^2 \tag{5.36}$$

where $k$ is from the set $\{1,2, \ldots, K\}$, the operator $\boldsymbol{P}$ is defined as

$$\boldsymbol{P}\left(\bar{\mathbf{A}}_k; \mathbf{x}\right) = \begin{bmatrix} \bar{a}_{k,1}x_1 + \bar{a}_{k,2}x_2 + \bar{a}_{k,3} \\ \bar{a}_{k,4}x_1 + \bar{a}_{k,5}x_2 + \bar{a}_{k,6} \end{bmatrix} \tag{5.37}$$

and $\mathbf{v}(\mathbf{x})$ is the dense-motion vector at pixel $\mathbf{x}$ given by

$$\mathbf{v}(\mathbf{x}) = \begin{bmatrix} v_1(\mathbf{x}) \\ v_2(\mathbf{x}) \end{bmatrix} \tag{5.38}$$

where $v_1$ and $v_2$ denote the horizontal and vertical components, respectively. All sites without labels are assigned one according to the motion-compensation criterion, which assigns the label of the parameter vector that gives the best motion compensation at that site. This feature ensures more robust parameter estimation by

eliminating the outlier vectors. Several post-processing operations may be employed to improve the accuracy of the segmentation map.

The procedure can be repeated by estimating new seed model parameters over the regions estimated in the previous iteration. Furthermore, the number of clusters can be varied by splitting or merging of clusters between iterations. The $K$-means method requires a good initial estimate of the number of classes $K$. The Hough transform methods do not require this information but are more expensive.

### *Hough Transform Methods*

The Hough transform is a well-known clustering technique where the data samples "vote" for the most representative feature values in a quantized feature space. In a straightforward application of the Hough transform to optical-flow segmentation using the six-parameter affine-flow model (5.37), the six-dimensional feature space $a_1, \ldots, a_6$ is quantized to a number of sets (states) after the minimal and maximal values for each parameter are determined. Then, each flow vector $\mathbf{v}(\mathbf{x})$ votes for a set $k$ of quantized parameters that minimizes

$$\varepsilon^2(\mathbf{x}) = \varepsilon_1^2(\mathbf{x}) + \varepsilon_2^2(\mathbf{x})$$

where

$$\varepsilon_1(\mathbf{x}) = v_1(\mathbf{x}) - \bar{a}_{k,1}x_1 - \bar{a}_{k,2}x_2 - \bar{a}_{k,3}$$

$$\varepsilon_2(\mathbf{x}) = v_2(\mathbf{x}) - \bar{a}_{k,4}x_1 - \bar{a}_{k,5}x_2 - \bar{a}_{k,6}$$

The parameter sets that receive at least a predetermined amount of votes are likely to represent candidate motions. The number of classes $K$ and the corresponding parameter sets to be used in labeling individual flow vectors are hence determined. The drawback of this scheme is the significant amount of computation and memory requirements involved.

In order to keep the computational cost at a reasonable level, several modified Hough methods have been presented. The proposed simplifications include [Adi 85]:

1. Decomposition of the parameter space into two disjoint subsets $\{a_1, a_2, a_3\} \times \{a_4, a_5, a_6\}$ to perform two 3D Hough transforms.
2. A multi-resolution Hough transform, where at each resolution level the parameter space is quantized around the estimates obtained at the previous level.

3. A multi-pass Hough technique, where the flow vectors that are most consistent with the candidate parameters are grouped first. In the second stage, those components formed in the first stage that are consistent with the same flow model in the least-squares sense are merged together to form segments. Several merging criteria have been proposed. In the third and final stage, ungrouped flow vectors are assimilated into one of their neighboring segments.

Other simplifications that are proposed include the probabilistic Hough transform [Kir 91] and the randomized Hough transform [Kru 96].

Clustering in the parameter space has some drawbacks: i) both methods rely on pre-computed optical flow as input, which is generally blurred at motion boundaries and may contain outliers; ii) clustering based on distances in the parameter space can lead to clustered parameters that are not physically meaningful and the results are sensitive to the choice of the weight matrix **M** and small errors in the estimation of affine parameters; and iii) parameter-clustering and label-assignment procedures are decoupled; hence, ad-hoc post-processing operations that depend on some threshold values are needed to clean up the final segmentation map. The maximum-likelihood segmentation method, discussed next, addresses these shortcomings.

### Maximum-Likelihood Segmentation

Motion-segmentation approaches in general are classified as optical-flow segmentation methods, which operate on pre-computed optical-flow estimates, and direct methods, which operate on spatio-temporal intensity values. We present here a unified formulation that covers both cases. The ML method finds the segmentation labels that maximize the likelihood function, which models the deviation of the observations (estimated dense-motion vectors or observed intensity values) from a parametric description of them (parametric-motion vectors or motion-compensated intensity values, respectively) for a given motion model.

We start by defining the log-likelihood function as

$$L(\mathbf{o} \mid \mathbf{z}) = \log(p(\mathbf{o} \mid \mathbf{z})) \tag{5.39}$$

where $\mathbf{z}$ denotes the lexicographical ordering of the segmentation labels $z(\mathbf{x})$, which takes values from the set $\{1,2,\ldots,K\}$ at each pixel $\mathbf{x}$. The vector $\mathbf{o}$ stands for the lexicographic ordering of the observations, which are either estimated dense-motion (optical flow) vectors or image intensity values. The conditional probability $p(\mathbf{o}|\mathbf{z})$ quantifies how well piecewise parametric-motion modeling fits the observations

$\mathbf{o}$ given the segmentation labels $\mathbf{z}$. If we model the mismatch between the observations $\mathbf{o}(\mathbf{x})$ and their parametric representations computed by the operator $\boldsymbol{O}(\mathbf{A}_{z(\mathbf{x})};\mathbf{x})$,

$$\varepsilon = \mathbf{o}(\mathbf{x}) - \boldsymbol{O}(\mathbf{A}_{z(\mathbf{x})};\mathbf{x})$$

where $\mathbf{A}_k$ denotes the $k$th parametric-motion model, by white, Gaussian noise with zero-mean and variance $\sigma^2$, then the conditional pdf of the observations given the segmentation labels can be expressed as

$$p(\mathbf{o}\,|\,\mathbf{z}) = \frac{1}{(2\pi\sigma^2)^{M/2}}\, e^{-\frac{\sum_{i=1}^{M}\varepsilon^2(\mathbf{x}_i)}{2\sigma^2}} \tag{5.40}$$

where $M$ is the number of observations available at site $\mathbf{x}_i$. Assuming that the parametric-flow model is more or less accurate, this deviation is due to the presence of observation noise (given correct segmentation labels). Then, the problem is finding the $K$-motion models, $\mathbf{A}_1\ \mathbf{A}_2, \ldots, \mathbf{A}_K$, and a label field $z(\mathbf{x})$to maximize the log-likelihood function $L(\mathbf{o}|\mathbf{z})$. We consider two cases.

### *Case 1 – Pre-Computed Optical-Flow Segmentation*

The observation $\mathbf{o}(\mathbf{x})$ stands for the estimated dense-motion vectors $\mathbf{v}(\mathbf{x})$, and operator $\boldsymbol{O}$ stands for the parametric-motion operator $\boldsymbol{P}$ given by Eq. (5.37) or a higher-order model (see Section 4.2.3) given by

$$\tilde{v}_1(\mathbf{x}) = a_1x_1 + a_2x_2 + a_3 + a_7x_1^2 + a_8x_1x_2 \tag{5.41a}$$

$$\tilde{v}_2(\mathbf{x}) = a_4x_1 + a_5x_2 + a_6 + a_7x_1x_2 + a_8x_2^2 \tag{5.41b}$$

Then,

$$\varepsilon^2(\mathbf{x}_i) = [v_1(\mathbf{x}_i) - \tilde{v}_1(\mathbf{x}_i)]^2 + [v_2(\mathbf{x}_i) - \tilde{v}_2(\mathbf{x}_i)]^2 \tag{5.42}$$

is the norm-squared deviation of the actual flow vectors from what is predicted by the quadratic-flow model. This case concerns motion segmentation by motion-vector matching.

### *Case 2 – Direct Segmentation*

The observation $\mathbf{o}(\mathbf{x})$ stands for the scalar-pixel intensities $I_t(\mathbf{x})$ at frame $t$, and the operator $\boldsymbol{O}$ is the motion-compensation operator $\boldsymbol{Q}$, defined by

$$Q(\mathbf{A}_{z(\mathbf{x})};\mathbf{x}) = I_{t-1}(\mathbf{x}') \tag{5.43}$$

where

$$\mathbf{x}' = \begin{bmatrix} x_1' \\ x_2' \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + P(\mathbf{A}_{z(\mathbf{x})};\mathbf{x})$$

Then,

$$\varepsilon^2(\mathbf{x}_i) = \left(I_t(\mathbf{x}) - I_{t-1}(\mathbf{x}')\right)^2 \tag{5.44}$$

This case is segmentation by motion-compensated intensity matching. Motion parameters $\mathbf{A}_k$ are estimated over the support of model $k$ (see step 3 below).

In either case, assuming that the variances for all classes are the same, maximization of the log-likelihood function is equivalent to minimization of the cost function

$$\sum_{(\mathbf{x}_1,\mathbf{x}_2)} \left\|\mathbf{o}(\mathbf{x}) - O(\mathrm{A}_{z(\mathbf{x})};\mathbf{x})\right\|^2 \tag{5.45}$$

or equivalently

$$\sum_{k=1}^{K} \sum_{\mathbf{x}\epsilon Z_k} \left\|\mathbf{o}(\mathbf{x}) - O_k(\mathbf{x})\right\|^2$$

where $Z_k$ is the set of pixels $\mathbf{x}$ with motion label $z(\mathbf{x}) = k$ and $O_k(\mathbf{x}) = O(\mathbf{A}_k; \mathbf{x})$.

An iterative solution to this problem is given by

1. Initialize $\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_K$.
2. Assign a motion label $z(\mathbf{x})$ to each pixel $\mathbf{x}$ as

$$z(\mathbf{x}) = \arg\min_k \left\|\mathbf{o}(\mathbf{x}) - O(\mathbf{A}_k;\mathbf{x})\right\|^2 \tag{5.46}$$

   where $k$ takes values from the set $\{1, 2, \ldots, K\}$.
3. Update $\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_K$ as

$$\mathbf{A}_k = \arg\min_{\mathbf{A}} \sum\nolimits_{\mathbf{x}\epsilon Z_k} \left\|\mathbf{v}(\mathbf{x}) - P(\mathbf{A};\mathbf{x})\right\|^2 \tag{5.47}$$

   This minimization is equivalent to least-squares estimation of the affine-motion model fit to those motion vectors with the label $z(\mathbf{x}) = k$. A closed form solution to this problem can be expressed in terms of a linear matrix equation

$$\begin{bmatrix} x_1 & x_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & x_2 & 1 \end{bmatrix} \begin{bmatrix} a_{k,1} \\ a_{k,2} \\ a_{k,3} \\ a_{k,4} \\ a_{k,5} \\ a_{k,6} \end{bmatrix} = \begin{bmatrix} x_1 + v_1(\mathbf{x}) \\ x_2 + v_2(\mathbf{x}) \end{bmatrix} \tag{5.48}$$

for all $\mathbf{x}$ such that $z(\mathbf{x}) = k$.

4. Repeat steps 2 and 3 until the class means $\mathbf{A}_k$ do not change by more than a predefined amount between successive iterations.

This method does not require gradient-based optimization or other numeric search procedures for optimization of a cost function. Thus, it is robust and computationally efficient. Extensions using mixture modeling and robust estimators that require gradient-based optimization have also been proposed [Aye 95].

Motion-vector matching is a good segmentation criterion when the estimated motion field is accurate; i.e., if all outlier motion estimates are eliminated. Motion-compensated intensity matching is a more suitable criterion when spatial-intensity (color) variations are sufficient and/or a multi-resolution labeling procedure is employed.

A possible limitation of the ML segmentation framework is that it lacks constraints to enforce spatial and temporal continuity of the segmentation labels. Thus, ad-hoc procedures are needed to eliminate small, isolated regions in the segmentation label field. The MAP segmentation strategy promises to impose continuity constraints in an optimization framework.

### Maximum *A Posteriori* Probability Segmentation

The MAP method is a Bayesian approach that searches for the maximum of the *a posteriori* pdf of the segmentation labels given the observations (either estimated optical flow or observed intensity data). The *a posteriori* pdf is not only a measure of how well the segmentation labels explain the observed data, but also how well they conform to our prior expectations. The MAP formulation differs from the ML approach in that it includes smoothness terms to enforce spatial continuity of the output-motion segmentation map.

The *a posteriori* pdf $p(\mathbf{z}|\mathbf{o})$ of the segmentation label field $\mathbf{z}$ given the observed data $\mathbf{o}$ can be expressed, using the Bayes theorem, as

$$p(\mathbf{z} \mid \mathbf{o}) = \frac{p(\mathbf{o} \mid \mathbf{z})\, p(\mathbf{z})}{p(\mathbf{o})} \tag{5.49}$$

where $p(\mathbf{o}|\mathbf{z})$ is the conditional pdf of optical-flow vectors given the segmentation $\mathbf{z}$ and $p(\mathbf{z})$ is the *a priori* pdf of the segmentation labels. Notice that: i) $\mathbf{z}$ is a discrete-valued random vector with a finite sample space $\Omega$, and ii) the pdf $p(\mathbf{o})$ is constant with respect to segmentation labels and, hence, can be ignored for the purpose of computing $\mathbf{z}$. The MAP estimate, then, maximizes the numerator of (5.49) over all possible realizations of the segmentation label field $\mathbf{z} = \omega$ where pixel labels $\omega \in \Omega$.

Modeling of the conditional pdf $p(\mathbf{o}|\mathbf{z})$ through (5.40) and (5.42) or (5.44) has been discussed while presenting the ML method. The prior pdf is modeled by a Gibbs distribution, which effectively introduces local smoothness constraints on the segmentation. The form of the prior pdf is given by Eqn. (5.8) in Section 5.1.3. Prior constraints on the structure of the segmentation labels, such as spatial smoothness, can be specifed in terms of the clique potentials (defned in Appendix B). Temporal continuity of the labels can similarly be modeled [Mur 87].

Substituting (5.40) and (5.8) into (5.49) and taking the logarithm of the resulting expression, maximization of the *a posteriori* pdf can be performed by minimizing the cost function

$$E = \frac{1}{2\sigma^2} \sum_{i=1}^{M} \varepsilon^2(\mathbf{x}_i) + U(\boldsymbol{\omega}) \tag{5.50}$$

The first term describes how well the predicted data fit the actual measurements (estimated optical-flow vectors or observed image-intensity values) and the second term measures how well the segmentation conforms to our prior expectations.

Because the motion-model parameters corresponding to each label are not known *a priori*, MAP segmentation must alternate between estimation of model parameters and assignment of the segmentation labels to optimize the cost function (5.50). Murray and Buxton [Mur 87] were the first to propose a MAP segmentation method, where the optical flow was modeled by a piecewise quadratic-flow field (5.41) and the segmentation labels were assigned based on a simulated annealing (SA) procedure (see Appendix C). Given the estimated fow feld $\mathbf{v}$ and the number of independent-motion models $K$, MAP segmentation using the Metropolis algorithm can be summarized as follows:

1. Start with an initial labeling $\mathbf{z}$ of the optical-flow vectors. Calculate the model parameters $a_1, a_2, \ldots, a_8$ for each region using least-squares fitting (similar to Eqn. (5.48)). Set the initial temperature for SA.

2. Update the segmentation labels at each site $\mathbf{x}_i$ as follows:
   a. Perturb the label $z_i = z(\mathbf{x}_i)$ randomly.
   b. Decide whether to accept or reject this perturbation, based on the change $\Delta E$ in the cost function (5.50),

$$\Delta E = \frac{1}{2\sigma^2}\Delta\varepsilon^2(\mathbf{x}_i) + \sum_{\mathbf{x}_j \in N_{\mathbf{x}_i}} \Delta V_C\left(z(\mathbf{x}_i), z(\mathbf{x}_j)\right) \tag{5.51}$$

   where $N_{\mathbf{x}_i}$ denotes a neighborhood of site $\mathbf{x}_i$ and $V_C\left(z(\mathbf{x}_i), z(\mathbf{x}_j)\right)$ is given by Eqn. (5.9). The first term indicates whether or not the perturbed label is more consistent with the given flow field determined by the residual (5.42), and the second term reflects whether or not it is in agreement with the prior segmentation field model.
3. After all pixel sites are visited once, re-estimate the mapping parameters for each region based on the new segmentation-label configuration. Note that the order in which the sites are visited affects the result because the update at each site is dependent on the labels of neighboring sites.
4. Exit if a stopping criterion is satisfied. Otherwise, lower the temperature according to a predefined temperature schedule, and go to step 2.

We can make the following observations: i) The MAP method carries a high computational cost. ii) The procedure proposed by Murray–Buxton suggests performing the model parameter update (step 3 above), after each and every perturbation. We did not notice a significant difference in performance if the motion-parameter updates were done after all sites are visited once. iii) The method can be applied with any parametric-motion model, although the original formulation has been developed on the basis of the eight-parameter model.

In addition to its high computational cost, the pixel-based MAP method cannot guarantee that the estimated motion boundaries coincide with spatial color edges (object boundaries). We next present an alternative ML region labeling approach to address this problem.

### 5.3.3 Region-Based Motion Segmentation: Fusion of Color and Motion

Fusion of contrast/color and motion can yield more robust video segmentation. This section extends the pixel-based ML method to region-based ML motion segmentation, where the image is first segmented into homogeneous *color* regions, and each color region is assigned a single motion label. It is generally true that motion boundaries

coincide with color-segment boundaries, but not vice versa; i.e., color segments are almost always a subset of motion segments as illustrated in Figure 5.2. Therefore, one can first perform color segmentation to obtain a set of candidate motion segments, such that each region has a single motion. Other approaches for region definition include superpixels ($N \times N$ blocks) and mesh-based partitioning of frames. The region-based motion label assignment strategy facilitates obtaining spatially continuous segmentation maps that are more closely related to actual object boundaries, without the heavy computational burden of the pixel-based Markov random field (MRF) model approach.

We assume that a region-formation procedure (e.g., color segmentation using fuzzy $C$-means algorithm [Lim 90]) has been performed on each video frame. We let $C(\mathbf{x})$ denote the region map of a frame consisting of $M$ mutually exclusive and exhaustive regions and define $\boldsymbol{C}_m$ as the set of pixels $\mathbf{x}$ with the region label $C(\mathbf{x}) = m$, $m = 1, \ldots, M$.

We seek the motion-segmentation vector $\mathbf{z}$ (formed by lexicographic ordering of $z(\mathbf{x})$) and the affine parameter vectors $\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_K$ that best fit the dense-motion field, such that

$$\sum_{i=1}^{M} \sum_{\mathbf{x} \in \boldsymbol{C}_m} \left\| \mathbf{v}(\mathbf{x}) - \boldsymbol{P}(\mathbf{A}_{z(m)}; \mathbf{x}) \right\|^2 \tag{5.52}$$

is minimized. Here, $\boldsymbol{P}$ is an operator defined by Eqn. 5.37, $z(m)$ refers to the motion label of all pixels within $\boldsymbol{C}_m$ and takes one of the values 1, 2, …, $K$, and $\mathbf{v}(\mathbf{x})$ is the dense-motion vector at pixel $\mathbf{x}$ as defined by Eqn (5.38). The procedure is given by [Alt 98]:

1. Initialize the motion-segmentation map $\mathbf{z}$ by assigning a single motion label $k$, $k = 1, \ldots, K$ to each $\boldsymbol{C}_m$.
2. Update the parameter vectors $\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_K$ as

$$\mathbf{A}_k = \arg\min_{\mathbf{A}} \sum_{\mathbf{x} \in \boldsymbol{Z}_k} \left\| \mathbf{v}(\mathbf{x}) - \boldsymbol{P}(\mathbf{A}; \mathbf{x}) \right\|^2 \tag{5.53}$$

where $\boldsymbol{Z}_k$ is the set of pixels $\mathbf{x}$ with the label $z(\mathbf{x}) = k$. This minimization can be achieved by solving the linear matrix equation

$$\begin{bmatrix} x_1 & x_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & x_2 & 1 \end{bmatrix} \begin{bmatrix} a_{k,1} \\ a_{k,2} \\ a_{k,3} \\ a_{k,4} \\ a_{k,5} \\ a_{k,6} \end{bmatrix} = \begin{bmatrix} v_1(\mathbf{x}) \\ v_2(\mathbf{x}) \end{bmatrix} \tag{5.54}$$

for all $\mathbf{x} \in \boldsymbol{Z}_k$.

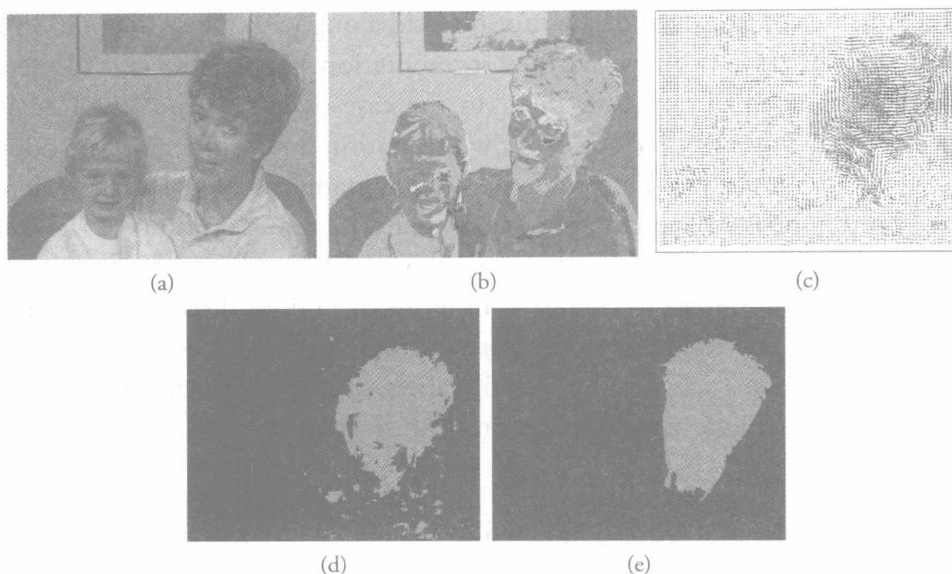



**Figure 5.2** Video segmentation: (a) a frame of "Mother and Daughter" sequence; (b) color-only pixel segmentation; (c) motion-vector field between two frames; (d) pixel-based motion segmentation; and (e) and color-region-based motion segmentation.

3. Assign a motion label to each region $\boldsymbol{C}_m$, $m = 1,2,\ldots,M$, such that

$$z(\boldsymbol{C}_m) = \arg\min_k \sum_{\mathbf{x}\in C_m} \left\| \mathbf{o}(\mathbf{x}) - \boldsymbol{O}(\mathbf{A}_k;\mathbf{x}) \right\|^2 \tag{5.55}$$

   where $k = 1,2, \ldots, K$ and $\mathbf{o}(\mathbf{x})$ and $\boldsymbol{O}(\mathbf{A}_k; \mathbf{x})$ are as defined in Section 5.3.2. This allows region-based affine motion segmentation with pixel-based motion-vector or intensity matching.
4. Repeat steps 2 and 3 until the class means $\mathbf{A}_k$ do not change by more than a pre-defined amount between successive iterations.

It can be clearly seen from Figure 5.2(e) that region-based label assignment results in a better segmentation of the head of the woman compared to pixel-based segmentation in Figure 5.2(d). We note that the pixel-based ML-motion segmentation presented in Section 5.3.2 is a special case of this region-based framework, where each region $\boldsymbol{C}_m$ contains a single pixel.

## 5.3.4 Simultaneous Motion Estimation and Segmentation

Until now, we have discussed methods to compute segmentation labels from either pre-computed optical flow or directly from intensity values, but have not addressed

how to compute an improved dense-motion field along with the segmentation map. It is clear that the success of optical-flow segmentation is closely related to the accuracy of the estimated optical-flow field (in the case of using pre-computed flow), and vice versa. It follows that optical-flow estimation and segmentation should be addressed simultaneously in a mutually beneficial manner for best results. Here, we present a simultaneous Bayesian approach based on a representation of the motion field as sum of a parametric field and a residual field. The interdependence of optical-flow and segmentation fields is expressed in terms of a Gibbs distribution within the MAP framework. The resulting optimization problem, to find estimates of a dense set of motion vectors, a set of segmentation labels, and a set of mapping parameters, is solved using the highest confidence first (HCF) and iterated conditional mode (ICM) algorithms.

### Motion-Field Model and MAP Framework

We model the optical-flow field $\mathbf{v}(\mathbf{x})$ as sum of a parametric-flow $\tilde{\mathbf{v}}(\mathbf{x})$ and a non-parametric residual field $\mathbf{v}_r(\mathbf{x})$ that accounts for local motion and other modeling errors, i.e.,

$$\mathbf{v}(\mathbf{x}) = \tilde{\mathbf{v}}(\mathbf{x}) + \mathbf{v}_r(\mathbf{x}) \tag{5.56}$$

The parametric component of the flow $\tilde{\mathbf{v}}(\mathbf{x})$ is calculated from the model parameters $\mathbf{A}_i$, $i = 1, \dots K$, which in turn is a function of $\mathbf{v}(\mathbf{x})$ and $z(\mathbf{x})$. The simultaneous MAP framework aims at maximizing the *a posteriori* pdf

$$p(\mathbf{v}_1, \mathbf{v}_2, \mathbf{z} \mid g_k, g_{k+1}) = \frac{p(g_{k+1} \mid g_k, \mathbf{v}_1, \mathbf{v}_2, \mathbf{z})\, p(\mathbf{v}_1, \mathbf{v}_2 \mid \mathbf{z}, g_k)\, p(\mathbf{z} \mid g_k)}{p(g_{k+1} \mid g_k)} \tag{5.57}$$

with respect to the optical-flow vectors $\mathbf{v}_1$, $\mathbf{v}_2$ and the segmentation labels $\mathbf{z}$, where $\mathbf{v}_1$ and $\mathbf{v}_2$ denote the lexicographic ordering of the first and second components of the flow vectors $\mathbf{v}(\mathbf{x}) = [v_1(\mathbf{x})\ v_2(\mathbf{x})]^T$ at each pixel $\mathbf{x}$. Through careful modeling of these pdfs, we can express an interrelated set of constraints that help improve both optical-flow and segmentation estimates.

The first term $p(g_{k+1} \mid g_k, \mathbf{v}_1, \mathbf{v}_2, \mathbf{z})$ in the numerator of (5.57) provides a measure of how well the present displacement and segmentation estimates conform to the observed frame $k+1$ given frame $k$. It is modeled by a Gibbs distribution as

$$p(g_{k+1} \mid g_k, \mathbf{v}_1, \mathbf{v}_2, \mathbf{z}) = \frac{1}{Q_1}\, e^{-U_1(g_{k+1} \mid g_k, \mathbf{v}_1, \mathbf{v}_2, \mathbf{z})} \tag{5.58}$$

where $Q_1$ is the partition function (normalizing constant) and

$$U_1(\boldsymbol{g}_{k+1} \mid \boldsymbol{g}_k, \mathbf{v}_1, \mathbf{v}_2, \boldsymbol{z}) = \sum_{\mathbf{x}} \left[ g_k(\mathbf{x}) - g_{k+1}(\mathbf{x} + \mathbf{v}(\mathbf{x})\Delta t) \right]^2$$

is called the Gibbs potential, which corresponds to the norm-square of the displaced frame difference (DFD) between the frames $g_k(\mathbf{x})$ and $g_{k+1}(\mathbf{x})$. Thus, maximization of (5.58) imposes that $\mathbf{v}(\mathbf{x})$ minimizes the DFD.

The second term is the conditional pdf of the displacement field given the motion segmentation and the search frame $k$. It is also modeled by a Gibbs distribution

$$p(\mathbf{v}_1, \mathbf{v}_2 \mid \boldsymbol{z}, \boldsymbol{g}_k) = p(\mathbf{v}_1, \mathbf{v}_2 \mid \boldsymbol{z}) = \frac{1}{Q_2} e^{-U_2(\mathbf{v}_1, \mathbf{v}_2 \mid \boldsymbol{z})} \tag{5.59}$$

where $Q_2$ is a constant and

$$U_2(\mathbf{v}_1, \mathbf{v}_2 \mid \boldsymbol{z}) = \alpha \sum_{\mathbf{x}} \left\| \mathbf{v}(\mathbf{x}) - \tilde{\mathbf{v}}(\mathbf{x}) \right\|^2 + \beta \sum_{\mathbf{x}_i} \sum_{\mathbf{x}_j \in N_{\mathbf{x}_i}} \left\| \mathbf{v}(\mathbf{x}_i) - \mathbf{v}(\mathbf{x}_j) \right\|^2 \delta(z(\mathbf{x}_i) - z(\mathbf{x}_j)) \tag{5.60}$$

is the corresponding Gibbs potential, $\|\cdot\|$ denotes the Euclidian distance, and $N_{\mathbf{x}}$ is the set of neighbors of site $\mathbf{x}$. The first term in (5.60) enforces a minimum norm estimate of the residual-motion field $\mathbf{v}_r(\mathbf{x})$; i.e., it aims to minimize the deviation of the optical-flow estimates $\mathbf{v}(\mathbf{x})$ from the parametric-motion field $\tilde{\mathbf{v}}(\mathbf{x})$ while minimizing the DFD. The second term in (5.60) imposes a piece-wise local smoothness constraint on the optical-flow estimates for those sites in the neighborhood $N_{\mathbf{x}}$ that has the same segmentation label with site $\mathbf{x}$. Thus, spatial smoothness is enforced only on the flow vectors within a single region. The parameters $\alpha$ and $\beta$ allow for relative scaling of the two terms.

The third term in ( 5.57) models the *a priori* probability of the segmentation field in a manner similar to explained in Section 5.1.3. It is given by

$$p(\boldsymbol{z} \mid \boldsymbol{g}_k) = p(\boldsymbol{z}) = \frac{1}{Q_3} \sum_{\omega \in \Omega} e^{-U_3(\boldsymbol{z})} \delta(\boldsymbol{z} - \boldsymbol{\omega}) \tag{5.61}$$

where $\Omega$ denotes the sample space of the discrete-valued random vector $\boldsymbol{z}$, and $Q_3$ and $U_3(\boldsymbol{z})$ are as defined in Eqn. (5.8). The dependence of labels on image intensity is usually neglected, although region boundaries generally coincide with intensity edges.

### Two-Step Iteration Algorithm

Maximizing the *a posteriori* pdf (5.57) is equivalent to minimizing the cost function

$$E = U_1(g_{k+1} \mid g_k, \mathbf{v}_1, \mathbf{v}_2, z) + U_2(\mathbf{v}_1, \mathbf{v}_2 \mid z) + U_3(\mathbf{z}) \tag{5.62}$$

that is composed of the potential functions in Eqs. (5.58), (5.59), and (5.61).

Direct minimization of (5.62) with respect to all unknowns is too difficult, because motion and segmentation fields contain a large number of unknowns. To this effect, we minimize (5.62) through the following two-step iteration [Cha 97]:

1. Given the best available estimates of the motion parameters $\mathbf{A}_i, i = 1, \ldots, K$, and $z$, update the optical-flow field $\mathbf{v}(\mathbf{x})$. This step involves the minimization of a modified cost function

$$E_1(\mathbf{v}(\mathbf{x})) = \sum_{\mathbf{x}} \left[ g_k(\mathbf{x}) - g_{k+1}(\mathbf{x} + \mathbf{v}(\mathbf{x})\Delta t) \right]^2 + \alpha \sum_{\mathbf{x}} \left\| \mathbf{v}(\mathbf{x}) - \tilde{\mathbf{v}}(\mathbf{x}) \right\|^2$$
$$+ \beta \sum_{\mathbf{x}_i} \sum_{\mathbf{x}_j \in N_{\mathbf{x}_i}} \left\| \mathbf{v}(\mathbf{x}_i) - \mathbf{v}(\mathbf{x}_j) \right\|^2 \delta\left( z(\mathbf{x}_i) - z(\mathbf{x}_j) \right) \tag{5.63}$$

which is composed of all terms in (5.62) that contain $\mathbf{v}(\mathbf{x})$. While the first term indicates how well the motion vectors $\mathbf{v}(\mathbf{x})$ explain observations, the second and third terms impose that they should conform to the parametric-flow model, and vary smoothly within each region, respectively. To minimize this energy function, we employ the HCF method proposed by Chou and Brown [Cho 90]. HCF is a deterministic method designed to efficiently handle optimization of multi-variable problems with neighborhood interactions.

2. Update the segmentation label field $z$, assuming that the optical-flow field $\mathbf{v}(\mathbf{x})$ is known. This step minimizes all terms in (5.62) that contain $z$ as well as $\tilde{\mathbf{v}}(\mathbf{x})$, given by

$$E_2(z) = \alpha \sum_{\mathbf{x}} \left\| \mathbf{v}(\mathbf{x}) - \tilde{\mathbf{v}}(\mathbf{x}) \right\|^2 + \sum_{\mathbf{x}_i} \sum_{\mathbf{x}_j \in N_{\mathbf{x}_i}} V_C(z(\mathbf{x}_i), z(\mathbf{x}_j)) \tag{5.64}$$

The first term in (5.64) quantifies the consistency of $\tilde{\mathbf{v}}(\mathbf{x})$ and $\mathbf{v}(\mathbf{x})$. The second term is the *a priori* probability of the present configuration of segmentation labels. We use the ICM procedure to optimize $E_2$ with respect to $z$ [Cha 97]. The mapping parameters $\mathbf{A}_i, i = 1, \ldots, K$ are updated by least-squares estimation within each region.

An initial estimate of the optical-flow field can be found by Bayesian estimation using a global smoothness constraint. Given this estimate, the segmentation labels can be initialized by a procedure similar to Wang and Adelson's [Wan 94]. The free parameters $\alpha, \beta$, and $\gamma$ may be chosen so that each term in the cost function (5.62) has equal emphasis. However, because the optimization is implemented in two steps, the ratio $\alpha/\gamma$ also becomes of consequence. It is recommended to select $1 \leq \alpha/\gamma \leq 5$, depending on how well the motion field can be represented by a piece-wise-parametric model and whether we have a sufficient number of classes to model the segmentation labels.

A hierarchical implementation, where $\mathbf{v}_1$, $\mathbf{v}_2$, and $\boldsymbol{z}$ can be estimated at different resolutions, is possible by constructing Gaussian pyramids of the images $\boldsymbol{g}_k$ and $\boldsymbol{g}_{k+1}$. The results of each hierarchy level are used to initialize the next level. Note that the Gibbs model for the segmentation labels has been extended to include neighbors in scale by Kato *et al.* [Kat 93].

Several other motion-analysis approaches can be formulated as special cases of this framework. If we retain only the first and third terms in (5.62), and assume that all sites possess the same segmentation label, then we have Bayesian-motion estimation with a global smoothness constraint. The motion-segmentation algorithm of Murray and Buxton [Mur 87] (Section 5.3.2) employs only the second term in (5.60) and third term in (5.62) to model the conditional and prior pdf, respectively. Wang and Adelson [Wan 94] rely on the first term in (5.60) to compute the motion segmentation (Section 5.3.2). However, they also take the DFD of the parametric motion vectors into consideration when the closest match between the estimated and parametric-motion vectors, represented by the second term, exceeds a threshold.

## 5.4 Motion Tracking

There are various technologies for motion tracking, including inertial sensing (e.g, accelerometers and gyroscopes), radio sensing (e.g., radio frequency identification (RFID) and global positioning system (GPS) tracking), vision-based methods, and hybrid methods that combine multiple sensors. In the following, we only discuss vision-based or visual tracking methods.

Visual-motion tracking computes temporally linked feature points (tracks or trajectories) or spatio-temporal segmentation maps (tubes) for one or more target objects in consecutive video frames by associating their appearance. Hence, tracking can be considered as causal spatio-temporal video segmentation, where color/motion-segmentation methods are extended to determine the map (template) of

an object in the current frame, given its map in the previous frame. The temporal association of templates can be difficult when the tracked object gets occluded, or its shape 3D orientation changes from frame to frame, or when it is moving too fast. Motion tracking may sometimes refer to estimating 3D-motion trajectory of a moving camera capturing a static scene, which is an important problem in robotic vision for self-navigation.

Various motion-tracking methods differ in how they model the appearance (color and/or shape) of moving objects and dynamics of motion. Some methods model objects by a cloud of points, some by fixed or adaptive templates, and others just model their contours. Hence, visual motion-tracking methods can be broadly classified as i) feature-point trackers, ii) template trackers, and iii) contour trackers. The temporal dynamics can be modeled in a prediction-update framework (e.g., particle filtering) or energy-minimization framework (e.g., active-contour modeling).

Feature-tracking methods can be classified as marker and markerless tracking methods. A set of fiducial markers with known color and size are often used in computer-vision applications, such as motion capture and augmented reality, to track the 3D pose of a marker coordinate system with respect to the camera coordinate system. Image of markers observed by the camera can be matched with the original known marker patterns. The pose of the marker with respect to the camera can be recovered using standard pose-estimation techniques. In the absence of markers, a set of image feature points, such as corner points, which can be automatically detected by means of image analysis, can be used for tracking.

Template-tracking methods employ a bounding box or an arbitrary-shaped template that can be tracked from frame to frame. The general idea is to project the current template into the next frame using color only (mean-shift or graph-based) or 2D-motion and color cues (KLT tracking or particle filtering discussed in Sections 5.4.2 and 5.4.4, respectively). The projected template can then be fine-tuned by morphological or other operators using color and edge information to obtain a more precise segmentation map to alleviate motion-estimation errors as well as include newly uncovered regions. Many algorithms employ fixed target/object appearance models, which are determined or trained before tracking begins and, hence, ignore changes in object texture and shape due to pose variations or lighting conditions during tracking. Some researchers address the template update problem or incremental learning of a low-dimensional object representation for more robust tracking in cluttered environments [Mat 04, Ros 08].

Active-contour trackers model and track connected contour segments by energy-minimization methods. We discuss specific tracking methods next.

### 5.4.1 Graph-Based Spatio-Temporal Segmentation and Tracking

Graph-based video segmentation is a direct extension of the graph-based image-segmentation methods discussed in Section 5.1.4. They form super-voxels, which are homogeneous space-time regions [Xu 12]. Grundmann *et al.* [Gru 10] generalize Felzenszwalb–Huttenlocher [Fel 04] graph-based segmentation to obtain an initial over-segmentation of a video volume into super-voxels by building a 3D graph. They use a tree structure to represent the segmentation hierarchy. They obtain regions that exhibit long-term temporal coherence by combining a volumetric over-segmentation with a hierarchical re-segmentation applying the same algorithm and using optical flow as a region descriptor for graph nodes. However, direct extensions of graph-based image segmentation methods are not causal, since they require access to the entire video sequence. To this effect, Grundmann *et al.* [Gru 10] proposed a clip-based processing approach to limit processing delay and memory usage.

A causal graph-based video segmentation has been proposed in [Cou 13], which first segments frame $k$ into super-pixels using the Felzenszwalb–Huttenlocher [Fel 04] method. Then, a graph is formed linking regions in frame $k-1$ (computed at the previous step) to super-pixels in frame $k$. Edge weights for links depend on the number of pixels in a region (frame $k-1$) and linked super-pixel (frame $k$), the difference of their mean color, and the distance between their centroids. The final segmentation for frame $k$ is computed by a minimum spanning forest procedure.

Graph-based video volume-segmentation methods are fully automatic and, unlike some tracking methods, do not require initialization or initial target detection. Hence, they may be used as pre-processing before some tracking applications that do not have strict real-time requirements.

### 5.4.2 Kanade–Lucas–Tomasi Tracking

In Chapter 4, we discussed Lucas and Kanade motion estimation [Luc 81], which is an iterative method to compute incremental displacements to register a template and an image. Later, Tomasi and Kanade [Tom 91] published a technical report, in which this method is used for tracking "good features" that satisfy certain criteria. In a later paper, Shi and Tomasi [Shi 94] proposed an additional step to verify that features are tracked correctly. Trackers, which follow a methodology based on these three papers and their extensions, are called Kanade–Lucas–Tomasi (KLT) trackers. KLT tracking can be used for feature-point tracking or template-based object tracking.

**Feature Tracking – Good Features to Track**

KLT can be used as a feature tracker by selecting "good feature points" based on a small template (e.g., $15 \times 15$) about each feature point [Tom 91]. A feature can be tracked reliably if a numerically stable solution to Eq. (4.32) can be found, which requires that the matrix $\mathbf{H}$ is well-conditioned. This is satisfied if the smaller eigenvalue is well above the noise level. That is, if $\lambda_1$ and $\lambda_2$ are eigenvalues of $\mathbf{H}$, the corresponding feature is a good feature to track if $\min(\lambda_1, \lambda_2) \gg \lambda$, where $\lambda$ is a threshold [Tom 91, Shi 94]. The main steps of the KLT feature tracker are:

1. Detect a set of feature points in the initial frame using a feature detector, such as Harris corner detection, where $\min(\lambda_1, \lambda_2) \gg \lambda$.
2. Find frame-to-frame correspondence vectors for each feature point using Lukas–Kanade motion estimation with a translation or affine-motion model (see Section 4.4.1) based on a local template about each feature point [Bak 04].
3. For each feature point, verify goodness of tracking at each frame. Some features can be removed (to eliminate those that are occluded or cannot be tracked accurately) and new ones may be added periodically (e.g., every five frames).

Shi and Tomasi [Shi 94] defined a good feature as a feature that can be tracked well over many frames without drift. To verify this, an affine transformation is fit between the image of the currently tracked feature and its image from a non-consecutive previous frame. If the affine-compensated image is too dissimilar the feature is dropped. When the current and reference templates are not related by an affine warping, the tracking residual $\epsilon_i$ between the $i$th template in the current and reference frames is an outlier, i.e., is not a sample from Gaussian distribution. Hence, the detection of bad features reduces to a problem of outlier detection, which is equivalent to estimating the mean and variance of a corrupted Gaussian distribution. Tommasini [Tom 98] proposed a simple model-free robust rejection rule, using median and median deviation instead of the mean and standard deviation. This rule prescribes to reject values that are more than $k$ median absolute deviations (MADs) away from the median, where

$$MAD = med_i \left\{ \left| \epsilon_i - med_j \, \epsilon_j \right| \right\}$$

A value of $k = 5.2$, under the hypothesis of Gaussian distribution, is adequate in practice, as it corresponds to about 3.5 standard deviations, and the range contains more than the 99.9% of a Gaussian distribution [Tom 98]. In order to limit the

adverse effects of slow intensity changes from frame-to-frame, the average gray level in the original and warped blocks can be subtracted in the computation of $\epsilon_i$.

**Template Tracking**

The KLT tracking framework has also been used for template tracking [Bak 04], where an example appearance of the object, extracted in the first frame as a template, is tracked in the remaining frames using parametric transformations (warping) of the template. An important challenge in template tracking is handling variations in object appearance during tracking that may be due to intrinsic factors, such as 3D-motion and/or shape deformation, or extrinsic causes, including illumination change, camera motion, and occlusions. A naive solution to this problem is to update the template every $n$ frames with a new image at the current template location. The problem with this naive approach is "drift" [Mat 04, Sch 07]. Each time the template is updated, small errors are introduced in the template location, which accumulate in time, and the template steadily drifts away from the object. A template-update algorithm that avoids drift has been proposed in [Mat 04], which retains multiple templates including the initial template from the first frame. The template is first updated with the image of the object at the current template location, which is then aligned with the initial template to compute the final updated template. An alternative sub-space projection framework, under "sub-space constancy assumption," called eigentracking has also been proposed for robust appearance-based tracking [Bla 98]. KLT tracking does not employ a temporal model of motion dynamics to enforce temporal consistency. This issue is addressed in Section 5.4.4.

### 5.4.3 Mean-Shift Tracking

While KLT requires a motion model, the MS algorithm (see Section 5.1.2) can be used for appearance-based (color only) tracking of objects (templates or blobs) where it is difficult to specify an explicit parametric-motion model. The target object is characterized by a color-histogram (pdf), $h_l, l = 1, \ldots, L$, where $L$ is the number of bins, which is computed within a $N \times N$ bounding box. We iteratively compute the location $\bar{\mathbf{x}}$ in the current frame that maximizes the Bhattacharyya coefficient

$$\rho(\bar{\mathbf{x}}) = \sum_{l=1}^{L} \sqrt{p_l(\bar{\mathbf{x}}) h_l} \tag{5.65}$$

between the histogram of the target template $\{h_l\}$ and the histogram $\{p_l(\bar{\mathbf{x}})\}$ of the bounding box centered at a location $\bar{\mathbf{x}}$ in the current frame. The maximum (mode)

of $\rho(\bar{\mathbf{x}})$ will be computed by using the MS algorithm, which consists of three main steps: histogram computation, weight calculation, and finding the new location. The complete algorithm is given by [Com 03]:

1. Select an $N \times N$ bounding box of a target (template) to be tracked in the initial frame (frame 0). Compute the $L$-bin color histogram $h_l, l = 1, \ldots, L$, of the template. Go to frame $j = 1$.
2. Set the MS iteration $k = 0$. Initialize the center $\bar{\mathbf{x}}_k^{(j)}$ of the bounding box.
3. Compute the color histogram $\left\{p_l\left(\bar{\mathbf{x}}_k^{(j)}\right)\right\}$ of the current bounding box.
4. Compute weights for each pixel $i = 1, \ldots, N^2$ within the bounding box

$$w_i(\bar{\mathbf{x}}_k^{(j)}) = \sum_{l=1}^{L} \delta[s(\mathbf{x}_i) - l] \sqrt{\frac{h_l}{p_l\left(\bar{\mathbf{x}}_k^{(j)}\right)}}$$

   where the Kronecker delta $\delta[s(\mathbf{x}_i) - l]$ denotes the histogram bin corresponding to color of the pixel $s(\mathbf{x}_i)$.
5. Update the center of the bounding box using the MS iteration

$$\bar{\mathbf{x}}_{k+1}^{(j)} = \frac{\sum_{i=1}^{N^2} \mathbf{x}_i w_i\left(\bar{\mathbf{x}}_k^{(j)}\right)}{\sum_{i=1}^{N^2} w_i\left(\bar{\mathbf{x}}_k^{(j)}\right)}$$

6. If $\left\|\bar{\mathbf{x}}_{k+1}^{(j)} - \bar{\mathbf{x}}_k^{(j)}\right\| < \delta$, increment the frame counter $j = j + 1$ and go to step 2. Otherwise, set $k = k + 1$ and go to step 3 to continue MS iterations.

In step 5, weighting using the Epanechnikov kernel $g(\mathrm{x}) = -\mathrm{k}'(\mathrm{x})$ may be used as

$$\bar{\mathbf{x}}_{k+1}^{(j)} = \frac{\sum_{i=1}^{N^2} \mathbf{x}_i\, w_i\left(\bar{\mathbf{x}}_k^{(j)}\right) g\left(\left\|\frac{x_j - x_i}{h}\right\|^2\right)}{\sum_{i=1}^{N^2} w_i\left(\bar{\mathbf{x}}_k^{(j)}\right) g\left(\left\|\frac{x_j - x_i}{h}\right\|^2\right)}$$

In some cases, the size of the target varies from frame-to-frame as it may move toward or away from the camera. Then, we need to adapt the size of the kernel (window) to obtain the best results. It has been proposed to run the algorithm three times, with window size $N_{new} = N$, $N_{new} = nint(1.1N)$, and $N_{new} = nint(0.9N)$, and then choose the best result [Com 03].

## 5.4.4 Particle-Filter Tracking

Particle filter is a Monte Carlo method used to compute a Bayesian state estimate recursively by updating *a posteriori* pdf of the state at each time $k$ based on all available information up to $k$. It is based on a probabilistic state-space formulation that requires a system model describing the evolution of the state with time and an observation model relating noisy measurements to the state at each time $k$. An optimal Bayesian estimate of the state and a measure of accuracy of the estimate may be obtained from the *a posteriori* pdf. Note that if the system and observation models are linear, and pdfs are Gaussian (e.g., clutter-free), the optimal recursive Bayesian estimate is given by the Kalman filter. When we have a non-linear system and/or non-Gaussian noise (e.g., cluttered background), it is often not possible to write closed-form expressions for the conditional and *a priori* pdfs (hence, the *a posteriori* pdf). Thus, particle filtering extends Kalman filtering to the case of non-linear and non-Gaussian models.

The key idea of particle filtering is to represent probability distributions by a weighted sample set (particles) [Aru 02]. The particle filter takes a large number of particles to represent the underlying distribution and updates particles at each time $k$, which approaches updating the *a posteriori* pdf as the number of particles goes to infinity; hence, the particle filter approaches the optimal Bayesian estimator. In template tracking, each particle represents a particular guess for the location of the object (template) tracked. The set of particles with more weight shows locations where the object is more likely to be. This weighted distribution is propagated through time, and we can determine the template trajectory by taking the particle with the highest weight or the weighted mean of the particle set at each time step.

Particles are sampled randomly from the prior probability distribution $p(\mathbf{x}_k)$ of the state vector $\mathbf{x}_k$. Each state vector typically consists of coordinates of a pixel and its motion vector and color attributes within a local neighborhood, which can be a rectangle or ellipse centered at the pixel. The number of particles varies between 50 and 500. The evolution of the set of particles from frame-to-frame is described by propagating each particle according to a dynamic system model, such as a constant velocity or constant acceleration model. Each particle is then weighted according to the conditional distribution $p(\boldsymbol{Z}_k|\mathbf{x}_k)$, where $\boldsymbol{Z}_k = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_k\}$ denotes all observations up to time $k$. The observation model measures an attribute of the object appearance, such as the color histogram (similar to MS tracking). Last, the mean state of the object is estimated at each time step. The re-sampling step allows replacing occluded

pixels with some newly uncovered pixels within the tracked object. The algorithm can be summarized step-by-step as [Num 03, Bra 07]:

1. **Initialization:** Select a bounding shape for the object, sample $N$ random particles $\{\mathbf{x}_0^{(l)}\}_{l=1}^{N}$ from *a priori* distribution $p(\mathbf{x}_0)$ favoring pixels in/near the box [Aru 02]. Set weights $\{W_0^{(l)} = 1/N\}_{l=1}^{N}$. For each frame $k = 0, 1, 2, \ldots$.
2. **Prediction (Density Propagation):** Propagate each particle $\mathbf{x}_{k+1}^{(l)} \sim p\left(\mathbf{x}_{k+1} \mid \mathbf{x}_k^{(l)}\right)$, $l = 1, \ldots, N$, given the dynamic model.
3. **Update**
   a. Given the observation $z_{k+1}$, compute the weights $W_{k+1}^{(l)} \propto W_k^{(l)} L\left(z_{k+1} \mid \mathbf{x}_{k+1}^{(l)}\right)$ for each particle $\mathbf{x}_{k+1}^{(l)}$ based on the likelihood $L\left(z_{k+1} \mid \mathbf{x}_{k+1}^{(l)}\right)$, which is a function of $p\left(z_{k+1} \mid \mathbf{x}_{k+1}^{(l)}\right)$.
   b. Normalize the weights, $\hat{W}_{k+1}^{(l)} = \dfrac{W_{k+1}^{(l)}}{\sum_{l=1}^{N} W_{k+1}^{(l)}}$.
   c. The state estimate $\hat{\mathbf{x}}_{k+1}$ is the weighted average of propagated particles
   $$\hat{\mathbf{x}}_{k+1} = \sum_{l=1}^{N} \hat{W}_{k+1}^{(l)} \mathbf{x}_{k+1}^{(l)}$$
   d. In order to suppress particles with low weight estimate the effective number of particles
   $$N_{eff} = \frac{1}{\sum_{l=1}^{N} \left(\hat{W}_{k+1}^{(l)}\right)^2}$$
   If $N_{eff} \leq N_{thres}$ then perform resampling; otherwise, increment frame $k$ and go to step 2.
4. **Resampling**
   a. Apply the resampling algorithm given in [Aru 02].
   b. Set new weights $W_{k+1}^{(l)} = \hat{W}_{k+1}^{(l)} = 1/N$, $l = 1, \ldots, N$.

Particle filtering provides a robust tracking framework, as it considers multiple state hypotheses simultaneously. Since less likely states have a chance to temporarily remain in the tracking process, particle filters can deal with short-lived occlusions.

Periodic analysis of the template (in MS tracking) or the observation model (in particle filtering) for object appearance variations is required in order to avoid drift. In particular, an incremental learning method, using the sequential Karhunen–Loeve (SKL) algorithm, which efficiently learns and updates a low-dimensional sub-space representation of the target object, has been proposed within the context of particle filtering with an affine-motion model [Ros 08]. They observe that at any time instance, it suffices to use an eigenbasis to account for appearance variation if the object-motion or illumination change is gradual.

### 5.4.5 Active-Contour Tracking

Contour-tracking methods model propagation of active-contour models from frame-to-frame. We can classify contour-tracking methods as deterministic-search-based methods and probabilistic (conditional-density propagation) methods.

#### Condensation (Conditional-Density Propagation)

The condensation algorithm [Isa 98] is a sample-based Monte Carlo method, where the conditional-density propagation concept behind the particle filtering (Section 5.4.4) was first proposed for detection and tracking of contours of a moving object in a cluttered environment. It is proposed to track curves in cluttered background. Hence, each sample (particle) $\left\{\mathbf{x}_k^{(l)}\right\}_{l=1}^{N}$ is a curve of varying position and shape (instead of a single pixel as in Section 5.4.4) with a thickness proportional to the weight $W_k^{(l)}$, and the weighted mean of these curves is computed as the state estimate. Compared to Kalman filtering, the condensation method is simpler and more general.

#### Motion Snake

Motion snake is a deterministic method that extends the discrete snake formulation that was introduced in Section 5.1.5 to fit an active-contour model to a desired object by search-based minimization of different energy functions. This section introduces new energy functions to model motion of an active contour from frame-to-frame [Fu 00, Par 01].

We observe that there are two different motions at the opposite sides of a motion boundary. This is illustrated in Figure 5.3 where the head of the dancer moves down while her arms move up. To this effect, the contour is divided into a number of segments such that there will be two candidate affine models, one for each side, for each contour segment. We estimate motion vectors inside and outside the contour on selected pixels along the angular bisectors at each node point as depicted in

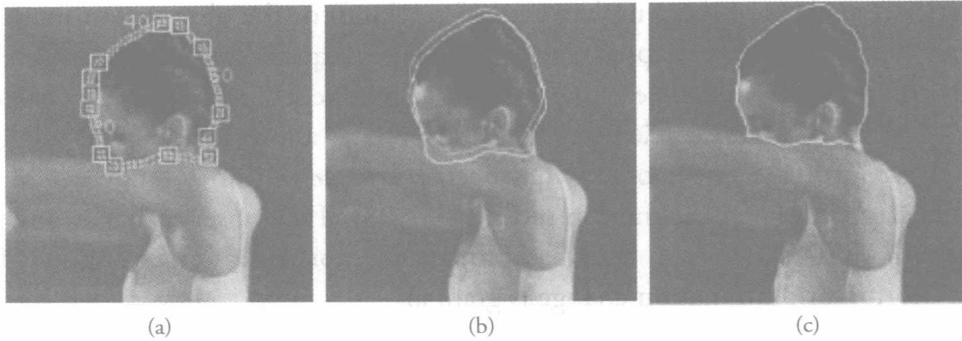



**Figure 5.3** Active-contour tracking: (a) contour in frame $k$; (b) two candidate predicted contour locations in frame $k+1$ based on inside and outside motion models; and (c) the contour in frame $k+1$ that minimizes the prediction energy [Fu 00]. (©IEEE 2000)

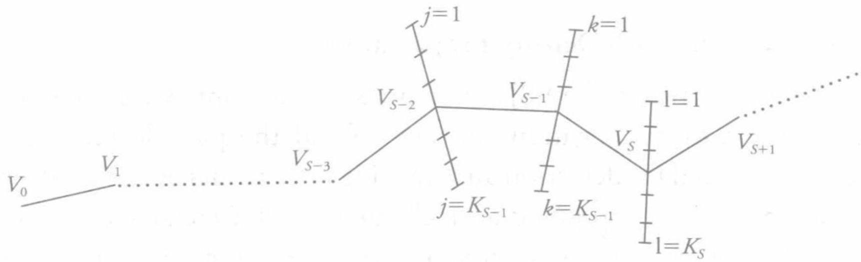



**Figure 5.4** Estimation of motion vectors inside and outside the contour [Fu 00]. (©IEEE 2000)

Figure 5.4. Once we obtain two affine motion models for each segment computed in two passes using estimated motion vectors inside and outside the contour segment, respectively, each segment has two candidate predicted locations in the next frame as depicted in Figure 5.3(b). In order to determine the correct predicted location for a segment, we compute a prediction energy for both predicted locations, and select the segment location with the smaller prediction energy with a bias favoring the location predicted by the motion vectors on the inside of the contour.

The complete motion-snake algorithm consists of the following basic steps [Fu 00]:

1. Specify an initial object contour in the first frame by marking a set of nodes along the desired contour to define an approximate polygonal shape, and then snap the approximate contour to fit the desired object tightly by minimizing intraframe energy terms.

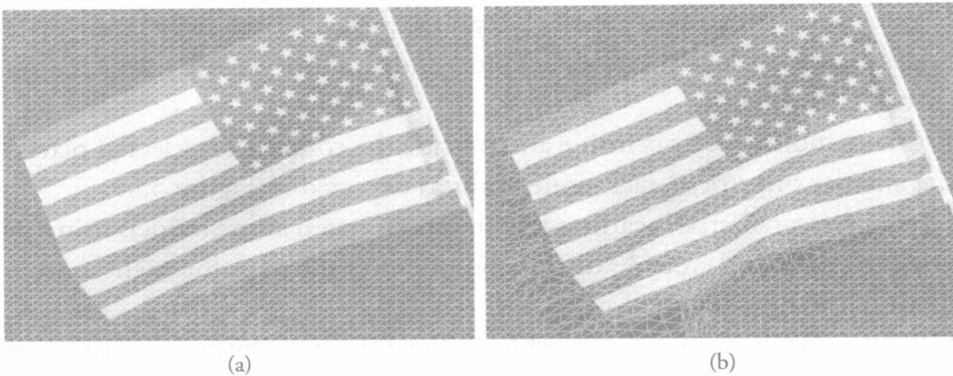
2. Segment the snake into non-overlapping pieces by selecting a set of feature nodes based on local curvature, color, and motion vectors using the method specified in [Fu 00].
3. Estimate motion vectors inside and outside the contour on selected pixels along the angular bisector at each node point as depicted in Figure 5.4.
4. Obtain two predicted locations (based on estimated affine models using motion vectors inside and outside of the contour, respectively) for each contour segment in the next frame.
5. Select one of these two predicted locations for each contour segment based on minimization of prediction energy.
6. Refine the predicted contour using interframe and intraframe energy terms, and go to step 2 to start processing the next frame.

In other related work, robust tracking of deformable models over long video sequences has been addressed in [Ker 94].

### 5.4.6 2D-Mesh Tracking

2D-mesh tracking is intended for tracking objects with mildly deformable motion without occlusions. A 2D mesh is a planar graph that tessellates (partitions) an image region into polygonal patches. The vertices of the patches are called node points. Patches are typically triangles or quadrangles, leading to triangular or quadrilateral meshes, respectively. Mesh-based motion models differ from block-based models in that patches overlap neither in the reference frame nor in the current frame (see Figure 5.5). Instead, triangular/polygonal patches in the current frame are deformed by the movements of the node points into respective patches in the reference frame, and texture within each patch in the reference frame is warped onto the current frame using a parametric model as a function of the node-point motion vectors [Tek 98].

The process of warping textures from one frame to another is called texture mapping. The affine model is used for texture mapping in triangular meshes. If proper constraints are imposed for parameter estimation, affine mapping guarantees continuity of motion and texture across triangle boundaries. This implies that the 2D-motion field can be compactly represented by the motion of node points, from which a continuous, piece-wise affine motion field can be reconstructed. The mesh structure constrains movements of adjacent image patches. Hence, meshes are well suited to represent mildly deformable but spatially continuous motion fields [Tok 96]. However, they do not allow motion discontinuities unless special constraints are applied to break the mesh structure at motion/occlusion boundaries [Alt 97].



(a) (b)

**Figure 5.5** 2D-mesh tracking: (a) initial uniform mesh and (b) motion-deformed mesh.

## 5.5 Image and Video Matting

Matting refers to accurate foreground object/subject estimation in images and video. It is a key technique to facilitate image and video editing or create novel composites for both professional film-production and consumer applications. An image $s(\mathbf{x})$ can be represented by a convex combination of a foreground $f(\mathbf{x})$ and background $b(\mathbf{x})$

$$s(\mathbf{x}) = \alpha\, f(\mathbf{x}) + (1-\alpha)\; b(\mathbf{x})$$

where $\alpha$ matte can take any value in the range [0,1]. If we consider the special case where alpha is binary, i.e., alpha values are only 0 or 1, the matting is equivalent to classic image/video segmentation, where each pixel belongs to either foreground or background. Matting is a more difficult problem than the classic segmentation, since we require extraction of semantically meaningful and pixel-accurate foreground objects together with the corresponding alpha values.

A semantic object may contain multiple colors, textures, motions, and shape deformations. Furthermore, the definition of semantic objects may depend on the context, which may not be captured by low level features. Hence, one should not expect to achieve semantically meaningful object segmentation using fully automatic methods based only on low-level features such as color, texture, shape, and motion. In general, extraction of semantically meaningful objects requires capture-specific information (e.g., chroma-keying) or user interaction. In matting, it is assumed that a tri-level image segmentation, called a *trimap*, which marks each pixel as definite foreground, definite background, or unknown is pre-specified by the user. Then, the matting problem is only solved for those pixels marked unknown to ensure extraction of the desired foreground [Wan 07].

### Chroma-Keying

Chroma-keying, also known as blue-screen matting, is a video-capture technology where each video object is recorded individually in a special studio against a key color, e.g., blue. The key color is selected such that it does not appear on the object to be captured. Then, the problem of extracting the object from each frame of video becomes one of color segmentation. Chroma-keyed video capture requires special attention to avoid shadows and other non-uniformity in the key color within a frame; otherwise, segmentation of key color may become a nontrivial problem.

### Interactive Semi-Automatic Segmentation

Since chroma-keying requires a special studio and/or equipment, a more practical alternative is interactive segmentation using *user interfaces* to aid a human operator. While background subtraction or motion segmentation may result in semantically meaningful objects in well-constrained settings, in an unconstrained environment, user interaction is indeed the only way to define a semantically meaningful object unambiguously because only the user can know what is semantically meaningful in a specific context. For example, if a person is running with a ball, whether the ball and the person are two separate objects or a single object may depend on the context.

### Image Matting

Image-matting methods can be classified as i) color-sampling methods, including Bayesian matting and the knockout algorithm; ii) affinity-based methods, which model the matte gradient, including Poisson matting, random-walk matting, closed-form matting; and iii) a combination of both, including robust and geodesic matting. The reader is referred to [Wan 07] for a discussion and comparison of these methods.

### Video Matting

We assume that the contour of a semantic object of interest is roughly sketched by an electronic pen or just marked by some feature points along its contour in selected *key frames* by a human operator. The approximate initial contours are then snapped tightly to the desired object automatically using, for example, the snake method. Once the precise boundary of the object of interest is determined in one or more keyframes, its boundary in all remaining frames can be automatically computed by motion tracking, e.g., by active-contour tracking (see Section 5.4.5). Finally, we allow a band of "unknown" pixels of desired width along the tracked contours, where high-precision matting problem is solved for these unknown pixels frame-by-frame.

## 5.6 Performance Evaluation

Comparative assessment of segmentation and tracking results is often based on subjective judgement, which is qualitative and time consuming. Hence, many studies have been performed to associate an objective figure of merit with image/video-segmentation and visual tracking results. We can classify these studies as those that require ground truth (GT) data and those that don't.

In practical applications, GT data is rarely available. However, a number of databases with ground truth are available for benchmarking studies. The Berkeley segmentation dataset contains more than 10,000 hand-labeled segmentations for benchmarking image segmentation and boundary detection [Mar 01]. The open development environment for evaluation of video systems (ODViS) [Jay 02] allows a user to generate GT data for pre-recorded video. Measures that do not rely on GT data generally evaluate intra-region homogeneity, inter-region disparity, and spatial or spatio-temporal consistency of results.

A set of performance measures that do not rely on GT data have been proposed in [Cor 03] that evaluate intra-object homogeneity based on shape regularity, spatial uniformity, temporal stability, and motion consistency. The inter-object disparity measures include local color and motion differences with neighboring regions. The usefulness of these measures has been demonstrated based on how the results predicted by these measures correlate with judgments of human observers. In another study [Erd 04], spatial-color contrast along the estimated object boundary, motion difference along the object boundary, and color-histogram difference between successive object segmentation masks in the temporal direction were evaluated. These measures can be computed per object and per frame, so that it is possible to identify the objects and frames that are poorly segmented within a long video.

Wu *et al.* [Wu 13] provide 50 fully annotated video sequences and 29 tracking algorithms for benchmarking of on-line visual-tracking algorithms. As an alternative to creating GT for videos, Black *et al.* [Bla 03] generate pseudo-synthetic video from a set of already compiled GT tracks to evaluate performance of tracking algorithms. Empirical standalone methods have been proposed to evaluate performance of tracking algorithms without using ground-truth data [Wu 10, San 12]. The framework in [San 12] is divided into two stages: i) estimation of the tracker condition to identify intervals during which a target is lost, and ii) measurement of the quality of the estimated track when the tracker is successful. Successful tracking is identified by analyzing the uncertainty of the tracker, whereas track recovery from errors is determined based on the time-reversibility constraint. Finally, Xu and Corso [Xu 12] discuss

what makes a good super-voxel segmentation method and evaluate the performance of some segmentation methods.

## References

[Aac 93] T. Aach, A. Kaup, and R. Mester, "Statistical model-based change detection in moving video," Signal Proc., vol. 31, no. 2, pp. 165–180, March 1993.

[Adi 85] G. Adiv, "Determining 3-D motion and structure from optical flow generated by several moving objects," IEEE Trans. on Patt. Anal. Mach. Intel., vol. 7, pp. 384–401, 1985.

[Alt 97] Y. Altunbasak and A. M. Tekalp, "Occlusion-adaptive content-based 2-D mesh design and tracking for object-based coding," IEEE Trans. on Image Processing, vol. 6, no. 9, pp. 1270–1280, Sep. 1997.

[Alt 98] Y. Altunbasak, E. Eren, and A. M. Tekalp, "Region-based affine motion segmentation using color information," Graph. Models and Image Proc., vol. 60, no. 1, pp. 13–23, Jan. 1998.

[Ami 90] A. A. Amini, T. E. Weymouth, and R. C. Jain, "Using dynamic programming for solving variational problems in vision," IEEE Trans. on Pattern Anal. Machine Intel., vol. 12, pp. 885–867, Sept. 1990.

[Aru 02] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for on-line nonlinear/nonGaussian Bayesian tracking," IEEE Trans. on Signal Processing, vol. 50, no. 2, pp. 174–188, Feb. 2002.

[Aye 95] S. Ayer and H. Sawhney, "Layered representation of motion video using robust maximum-likelihood estimation of mixture models and MDL coding," IEEE Int. Conf. Comp. Vision, Cambridge, MA, June 1995.

[Bak 04] S. Baker and I. Matthews, "Lucas-Kanade 20 years on: A unifying framework," International Journal of Computer Vision, vol. 56, no. 3, pp 221–255, Feb. 2004.

[Bar 11] O. Barnich and M. V. Droogenbroeck, "ViBe: A universal background subtraction algorithm for video sequences," IEEE Trans. Image Proc., vol. 20, no. 6, pp. 1709–1724, Jun. 2011.

[Ber 91] J. R. Bergen, P. J. Burt, K. Hanna, R. Hingorani, P. Jeanne, and S. Peleg, "Dynamic multiple-motion computation," in Artificial Intelligence and Computer Vision (Y.A. Feldman and A. Bruckstein, eds.), Elsevier, 1991, pp. 147–156.

[Bla 98] M. J. Black and A. D. Jepson, "EigenTracking: Robust matching and tracking of articulated objects using a view-based representation," Int. Jour. of Comp. Vision, vol. 26, no. 1, pp. 63–84, 1998.

[Bla 03] J. Black, T. Ellis, and P. Rosin, "A novel method for video tracking performance evaluation," in Joint IEEE Int. Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS), pp. 125–132, 2003.

[Bou 99] P. Bouthemy, M. Gelgon, and F. Ganansia, "A unified approach to shot change detection and camera motion characterization," IEEE Trans. on Circ. Syst. for Video Tech., vol. 9, pp. 1030–1044, Oct. 1999.

[Bra 07] P. Brasnett, L Mihaylova, D. Bull, and N. Canagarajah, "Sequential Monte Carlo tracking by fusing multiple cues in video sequences," Image and Vision Computing, vol. 25, no. 8, pp. 1217–1227, 2007.

[Bur 91] P. J. Burt, R. Hingorani, and R. Kolczynski, "Mechanisms for isolating component patterns in the sequential analysis of multiple motion," in IEEE Workshop on Visual Motion, pp. 187–193, Princeton, New Jersey, Oct. 1991.

[Cas 98] R. Castagno, T. Ebrahimi, and M. Kunt, "Video segmentation based on multiple features for interactive multimedia applications," IEEE Trans. on Circ. Syst. for Video Tech, vol. 8, no. 5, pp. 562–571, Sept. 1998.

[Cha 01] T. F. Chan and L. A. Vese, "Active contours without edges," IEEE Trans. on Image Proc., vol. 10, no. 2, pp. 266–277, Feb. 2001.

[Cha 97] M. M. Chang, A. M. Tekalp, and M. I. Sezan, "Simultaneous motion estimation and segmentation," IEEE Trans. on Image Processing, vol. 6, no. 9, pp. 1326–1333, Sep. 1997. (Also in Proc. ICASSP'94, Adelaide, Australia.)

[Che 95] Y. Cheng, "Mean shift, mode seeking, and clustering," IEEE Trans. on Pattern Anal. Machine Intel., vol. 17, no. 8, pp. 790–799, Aug. 1995.

[Chi 02] S.-Y. Chien, S.Y. Ma, and L.-G. Chen, "Efficient moving object segmentation algorithm using background registration technique," IEEE Trans. Circ. Syst. for Video Tech., vol. 12, no. 7, pp. 577–586, July 2002.

[Cho 90] P. B. Chou and C. M. Brown, "The theory and practice of Bayesian image labeling," Int. J. Comp. Vision, vol. 4, pp. 185–210, 1990.

[Col 79] G. B. Coleman and H. C. Andrews, "Image segmentation by clustering," Proc. of the IEEE, vol. 67, pp. 773–785, May 1979.

[Com 02] D. Comaniciu and P. Meer, "Mean shift: A robust approach towards feature space analysis," IEEE Trans. on Patt. Anal. Mach. Intel., vol. 24, no. 5, May 2002.

[Com 03] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," IEEE Trans. on Patt. Anal. and Machine Intell., vol. 25, no. 5, pp. 564–577, May 2003.

[Cor 03] P. L. Correia and F. Pereira, "Objective evaluation of video segmentation quality," IEEE Trans. on Image Proc., vol. 12, no. 2, pp. 186–200, Feb. 2003.

[Cor 04] P. L. Correia and F. Pereira, "Classification of video segmentation application scenarios," IEEE Trans. on Circ. Syst. for Video Tech., vol. 14, no. 5, pp. 735–741, May 2004.

[Cou 05] T. Cour, F. Benezit, and J. Shi, "Spectral segmentation with multiscale graph decomposition," IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR), 2005.

[Cou 13] C. Couprie, C. Farabet, Y. LeCun, L. Najman, "Causal graph-based video segmentation," Proc. of IEEE Int. Conf. Image Proc. (ICIP), Melbourne, Australia, Sept. 2013.

[Dem 77] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," J. of the Royal Statistical Society: Series B, vol. 39, no. 1, pp. 1–38, Nov. 1977.

[Der 87] H. Derin and H. Elliott, "Modeling and segmentation of noisy and textured images using Gibbs random field," IEEE Trans. Patt. Anal. Mach. Intel., vol. 9, pp. 39–55, Jan. 1987.

[Dim 02] N. Dimitrova, H. Zhang, B. Shahraray, I. Sezan, T, Huang, and A. Zakhor, "Applications of video content analysis and retrieval," IEEE Multimedia, vol. 9, pp. 42–55, July-Sept. 2002.

[Eki 03] A. Ekin, A. M. Tekalp, R. Mehrotra, "Automatic soccer video analysis and summarization," IEEE Trans. on Image Proc., vol. 12, no. 7, pp. 796–807, July 2003.

[Erd 04] C. Erdem, B. Sankur, and A. M. Tekalp, "Performance measures for video object segmentation and tracking," IEEE Trans. Image Proc., vol. 13, no. 7, pp. 937–951, July 2004.

[Fel 04] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," Int. Jour. of Computer Vision, vol. 59, no. 2, pp. 167–181, Sept. 2004.

[Fig 02] M. A. T. Figueiredo and A. K. Jain, "Unsupervised learning of finite mixture models," IEEE Trans. on Pattern Anal. Mach. Intell., vol. 24, no. 3, pp. 381–396, March 2002.

[Fu 00] Y. Fu, A. T. Erdem, and A. M. Tekalp, "Tracking visible boundary of objects using occlusion adaptive motion snake," IEEE Trans. on Image Processing, vol. 9, no. 12, pp. 2051–2060, Dec. 2000.

[Gar 00] U. Gargi, R. Kasturi, and S. H. Strayer, "Performance characterization of video-shot change detection methods," IEEE Trans. on Circ. Syst. Video Tech., vol. 10, pp. 1–13, Feb. 2000.

[Gon 07] R. C. Gonzalez and R. E. Woods, *Digital Image Processing, Third Edition*, Prentice Hall, 2007.

[Gru 10] M. Grundmann, V. Kwatra, M. Han, and I. Essa, "Efficient hierarchical graph based video segmentation," Proc. of IEEE CVPR, 2010.

[Ham 95] A. Hampapur, R. Jain, T. E. Weymouth, "Production model based digital video segmentation," Multimedia Tools Appl., vol. 1, pp. 9–46, 1995.

[Han 02] A. Hanjalic, "Shot-boundary detection: Unraveled and resolved?" IEEE Trans. Circ. Syst. for Video Tech., vol. 12, pp. 90–105, Feb. 2002.

[Har 85] R. M. Haralick and L. G. Shapiro, "Survey: Image segmentation techniques," Comp. Vis. Graph Image Proc., vol. 29, pp. 100–132, 1985.

[Hsu 94] S. Hsu, P. Anandan, and S. Peleg, "Accurate computation of optical flow by using layered motion representations," Proc. Int. Conf. Patt. Recog., Jerusalem, Israel, pp. 743–746, Oct. 1994.

[Ira 94] M. Irani, B. Rousso, and S. Peleg, "Computing occluding and transparent motions," Int. J. Computer Vision, vol. 12, no. 1, pp. 5–16, 1994.

[Ira 98] M. Irani and P. Anandan, "A unified approach to moving object detection in 2D and 3D scenes," IEEE Trans. on Patt. Anal. Machine Intel., vol. 20, no. 6, pp. 577–589, June 1998.

[Isa 98] M. Isard and A. Blake, "Condensation - Conditional density propagation for visual tracking," Int. J. Computer Vision, vol. 29, no. 1, pp. 5–28, 1998.

[Izq 02] E. Izquierdo and M. Ghanbari, "Key components for an advanced segmentation system," IEEE Trans. on Multimedia, vol. 4, no. 1, pp. 97–113, March 2002.

[Jay 02] C. Jaynes, S. Webb, R. Matt Steele, and Q. Xiong, "An open development environment for evaluation of video surveillance systems," Proc. of Int. Workshop on Performance Evaluation of Tracking and Surveillance (PETS'2002), Copenhagen, June 2002.

[Jia 98] H. Jiang, A. Helal, A. K. Elmagarmid, and A. Joshi, "Scene change detection techniques for video databases," Multimedia Systems, vol. 6, pp. 186–195, 1998.

[Kae 02] P. KaewTraKulPong and R. Bowden, "An improved adaptive background mixture model for real-time tracking with shadow detection," in Video-Based Surveillance Systems (ed. P. Remagnino, et al.), pp. 135–144, Kluwer 2002.

[Kas 88] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," Int. J. Computer Vision, vol. 1, no. 4, pp. 321–331, 1988.

[Kat 93] Z. Kato, M. Berthod, and J. Zerubia, "Parallel image classification using multiscale Markov random fields," Proc. of the IEEE ICASSP, Minneapolis, MN, pp. V137–140, April 1993.

[Ker 94] C. Kervrann and F. Heitz, "Robust tracking of stochastic deformable models in long image sequences," Proc. IEEE Int. Conf. Image Proc., Austin, TX, Nov. 1994.

[Kir 91] N. Kiryati, Y. Eldar, and A.M. Bruckstein, "A probabilistic Hough transform," Pattern Recognition, vol. 24, no. 4, pp. 303–316, 1991.

[Kol 04] V. Kolmogorov and R. Zabih, "What energy functions can be minimized via graph cuts?" IEEE Trans. on Patt. Anal. and Mach. Intell., vol. 26, no. 2, pp. 147–159, Feb. 2004.

[Kop 01] I. Koprinska and S. Carrato, "Temporal video segmentation: A survey," Signal Proc.: Image Communication, vol. 16, no. 5, pp. 477–500, Jan. 2001.

[Kru 96] S.-M. Kruse, "Scene segmentation from dense displacement vector fields using randomized Hough transform," Signal Proc.: Image Comm., vol. 9, pp. 29–41, 1996.

[Lee 90] S. U. Lee, S. Y. Chung, and R. H. Park, "A comparative performance study of several global thresholding techniques for segmentation," Comp. Vis. Graph Image Proc., vol. 52, pp. 171–190, 1990.

[Lel 03] D. Lelescu and D. Schonfeld, "Statistical sequential analysis for real-time video scene change detection on compressed multimedia stream," IEEE Trans. on Multimedia, vol. 5, no. 1, pp. 106–117, March 2003.

[Lia 01] P.-S. Liao, T.-S. Chen, and P.-C. Chung, "A fast algorithm for multilevel thresholding," J. Inf. Sci. Eng., vol. 17, no. 5, pp. 713–727, 2001.

[Lie 01] R. Lienhart, "Reliable transition detection in videos: A survey and practitioner's guide," Int. J. Image Graph., vol. 1, pp. 469–486, Aug. 2001.

[Lim 90] Y. W. Lim and S. U. Lee, "On the color image segmentation algorithm based on the thresholding and the fuzzy c-means techniques," Patt. Recog., vol. 23, no. 9, pp. 935–952, 1990.

[Luc 81] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," Int. Joint Conf. on Artificial Intell., pp. 674–679, 1981.

[Mar 01] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," Proc. of IEEE Int. Conf. Computer Vision (ICCV), vol. 2, pp. 416–423, 2001.

[Mat 04] I. Matthews, T. Ishikawa, and S. Baker, "The template update problem," IEEE Trans. on Patt. Anal. and Machine Intell., vol. 26 no. 6, pp. 810–815, June 2004.

[Mec 98] R. Mech and M. Wollborn, "A noise robust method for 2D shape estimation of moving objects in video sequences considering a moving camera," Signal Processing, vol. 66, no. 2, pp. 203–217, April 1998.

[Mur 87] D.W. Murray and B.F. Buxton, "Scene segmentation from visual motion using global optimization," IEEE Trans. Patt. Anal. Mach. Intel., vol. 9, no. 2, pp. 220–228, Mar. 1987.

[Ner 98] A. Neri, S. Colonnese, G. Russo, P. Talone, "Automatic moving object and background separation," Signal Processing, vol. 66, no. 2, pp. 219–232, April 1998.

[Num 03] K. Nummiaro, E. Koller-Meier, and L. Van Gool, "An adaptive color-based particle filter," Image and Vision Computing, vol. 21, no. 1, pp. 99–110, Jan. 2003.

[Ots 79] N. Otsu, "A threshold selection method from gray-level histograms," IEEE Trans. on Systems, Man and Cybernetics, vol. 9, no. 1, pp.62–66, 1979.

[Pal 93] N. Pal and S. Pal, "A review on image segmentation techniques," Patt. Recog., vol. 26, no. 9, pp. 1277–1294, Sep. 1993.

[Pap 92] T. N. Pappas, "An adaptive clustering algorithm for image segmentation," IEEE Trans. on Signal Proc., vol. SP-40, pp. 901–914, Apr. 1992.

[Par 01] H. W. Park, T. Schoepflin, and Y. Kim, "Active contour model with gradient directional information: Directional snake," IEEE Trans. on Circ. Syst. Video Tech., vol. 11, no. 2, Feb. 2001.

[Rad 05] R. Radke, S. Andra, O. Al-Kofahi, and B. Roysam, "Image change detection algorithms: A systematic survey," IEEE Trans. on Image Proc., vol. 14, no. 3, pp. 294–307, Mar. 2005.

[Ros 08] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, "Incremental learning for robust visual tracking," Int. J. of Computer Vision, vol. 77, no. 1–3, pp. 125–141, May 2008.

[Sal 99] P. Salembier and F. Marques, "Region-based representations of images and video: Segmentation tools for multimedia services," IEEE Trans. on Circ. Syst. for Video Tech., vol. 9, no. 8, pp. 1147–1169, Dec. 1999.

[San 12] J. C. SanMiguel, A. Cavallaro, J. M. Martínez, "Adaptive online performance evaluation of video trackers," IEEE Trans. on Image Proc., vol. 21, no. 5, pp. 2812–2823, 2012.

[Saw 95] H. Sawhney, S. Ayer, and M. Gorkani, "Model-based 2D and 3D dominant motion estimation for mosaicing and video representation," IEEE Int. Conf. Computer Vision, Cambridge, MA, June 1995.

[Sch 07] D. Schreiber, "Robust template tracking with drift correction," Pattern Recognition Letters, vol. 28, no. 12, pp. 1483–1491, Sept. 2007.

[Sez 04] M. Sezgin and B. Sankur, "Survey over image thresholding techniques and quantitative performance evaluation," J. of Electronic Imaging, vol. 13, no. 1, pp. 146–165, 2004.

[Shi 94] J. Shi and C. Tomasi, "Good features to track," IEEE Conf. Comp. Vision and Patt. Recog. (CVPR), pp. 593–600, Seattle, WA, June 1994.

[Shi 00] J. Shi and J. Malik, "Normalized cuts and image segmentation," IEEE Trans. on Patt. Anal. and Mach. Intel., vol. 22, no. 8, pp. 888–905, Aug. 2000.

[Sta 99] C. Stauffer and W.E.L. Grimson, "Adaptive background mixture models for real-time tracking," Proc. of the IEEE Computer Vision and Pattern Recognition, vol. 2, Ft. Collins, CO, USA, June 1999.

[Str 00] E. Stringa and C. S. Regazzoni, "Real-time video shot detection for scene surveillance applications," IEEE Trans. on Image Proc., vol. 9, no. 1, pp. 69–79, Jan. 2000.

[Sun 02] H. Sundaram and S.-F. Chang, "Computable scenes and structures in films," IEEE Trans. on Multimedia, vol. 4, pp. 482–491, Dec. 2002.

[Tao 07] W. Tao, H. Jin, and Y. Zhang, "Color image segmentation based on mean shift and normalized cuts," IEEE Trans. on Syst., Man and Cyber.- Part B: Cybernetics, vol. 27, no. 9, pp. 1382–1389, Oct. 2007.

[Tek 98] A. M. Tekalp, P. J.L. van Beek, C. Toklu, and B. Gunsel, "2D mesh-based visual object representation for interactive synthetic/natural video," Proc. of the IEEE, vol. 86, no. 6, pp. 1029–1051, June 1998.

[Tok 96] C. Toklu, A. T. Erdem, M. I. Sezan, and A. M. Tekalp, "Tracking motion and intensity-variations using hierarchical 2-D mesh modeling for synthetic object transfiguration," Graphical Models and Image Processing, vol. 58, no. 6, pp. 553–573, Nov. 1996.

[Tom 91] C. Tomasi and T. Kanade, Detection and Tracking of Point Features. *Carnegie Mellon University Technical Report CMU-CS-91–132,* April 1991.

[Tom 98] T. Tommasini, A. Fusiello, E. Trucco, and V. Roberto, "Making good features track better," IEEE Conf. Comp. Vision and Patt. Recog. (CVPR), June 1998.

[Tsa 01] A. Tsai, A. R. Yezzi, A. S. Willsky, "Curve evolution implementation of the Mumford-Shah functional for image segmentation, denoising, interpolation, and magnification," IEEE Trans on Image Proc., vol. 10, no. 8, pp. 1169–1186, Aug. 2001.

[Wan 94] J. Y. A. Wang and E. Adelson, "Representing moving images with layers," IEEE Trans. on Image Proc., vol. 3, pp. 625–638, Sep. 1994.

[Wan 00] Y. Wang, Z. Liu, and J.-C. Huang, "Multimedia content analysis using both audio and video clues," IEEE Signal Proc. Magazine, vol. 17, pp. 12–36, Nov. 2000.

[Wan 07] J. Wang and M. F. Cohen, "Image and video matting: A survey," Foundations and Trends in Computer Graphics and Vision, vol. 3, no. 2, pp. 97–175, Jan. 2007

[Wei 96] Y. Weiss and E. H. Adelson, "A unified mixture framework for motion segmentation: incorporating spatial coherence and estimating the number of models," Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition, June 1996.

[Wu 93] S. F. Wu and J. Kittler, "A gradient-based method for general motion estimation and segmentation," J. Vis. Comm. Image Rep., vol. 4, no. 1, pp. 25–38, Mar. 1993.

[Wu 10] H. Wu, A. Sankaranarayanan, and R. Chellappa, "Online empirical evaluation of tracking algorithms," IEEE Trans. Patt. Anal. Mach. Intell., vol. 32, no. 8, pp. 1443–1458, 2010.

[Wu 13] Y. Wu, J. Lim, and M.-H. Yang, "On-line object tracking: A benchmark," in IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2013.

[Xu 12] C. Xu and J.J. Corso, "Evaluation of super-voxel methods for early video processing," ECCV 2012.

[Yeo 95] B. L. Yeo and B. Liu, "Rapid scene analysis on compressed videos," IEEE Trans. on Circ. Syst. Video Tech., vol. 5, no. 6, Dec. 1995.

[Zha 93] H. J. Zhang, A. Kankanhalli, and S. W. Smoliar, "Automatic partitioning of full-motion video," Multimedia Systems, vol. 1, pp. 10–28, 1993.

## MATLAB Exercises

**5.1** Background Subtraction

Given a sequence with at least 20 frames, start processing from frame 11:

**a.** Compute the running mean of the previous 10 frames.

**b.** Compute the median of the previous 10 frames.

**c.** Determine a suitable threshold value to detect moving objects by subtracting the mean- or median-filtered model frames from the current frame.

**d.** Display both the model frames and the difference frames showing moving objects. How do you compare mean vs. median filtering to compute model frames? Comment on the results.

5.2 KLT Tracker
Given a sequence with $N$ frames,
a. Determine an initial object template in the first frame. Track the fixed template by affine warping toward successive frames.
b. Select the same initial template. Track the template this time by updating the template every five frames.
c. Comment on how the tracker with or without template updates behaves in the presence of clutter or occlusion.

5.3 Mean-Shift Tracker
Given the same video sequence and the initial object template as in Exercise 2, implement the MS template tracker. Compare the results with those of Exercise 2 and comment on the performance of the two methods.

## Internet Resources

The Berkeley Segmentation Dataset and Benchmark
http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/

ViBe – Background Subtraction
http://www2.ulg.ac.be/telecom/research/vibe/

KLT: An Implementation of the Kanade–Lucas–Tomasi Feature Tracker
http://www.ces.clemson.edu/~stb/klt/

The Condensation Algorithm
http://www.robots.ox.ac.uk/~misard/condensation.html

Code and Data for Incremental Learning for Robust Visual Tracking
http://www.cs.toronto.edu/~dross/ivt/

Visual Tracker Benchmark
https://sites.google.com/site/trackerbenchmark/benchmarks/v10

CHAPTER 6

# Video Filtering

*The performance of single-frame (image) de-noising and restoration methods may be improved by multi-frame filtering, whereas video-format conversion and super-resolution reconstruction are inherently multi-frame filtering problems.*

This chapter extends the image-filtering methods covered in Chapter 3 by introducing new methods that are specific to video, including multiple-picture (field or frame) filtering methods. Multi-frame filters can be classified as linear spatio-temporal filters, motion-adaptive filters, and motion-compensated filters. The theory of linear spatio-temporal filtering is provided in Section 6.1. Temporal frequency content of a video is dependent on the spatial-frequency content of a key frame and its motion. Hence, video filters should be designed by taking the motion content of the video into account. Because errors in motion detection and motion estimation are unavoidable, a fallback mode that does not use motion information (intra-mode filtering) should always be supported for robust processing without visual artifacts. Motion-adaptive methods, which require motion detection, and motion-compensated filtering, which requires true motion estimation, are often designed specific to a problem. Multi-frame filters designed for video-format conversion, de-noising, restoration, and super-resolution are introduced in Sections 6.2 to 6.5, respectively.

## 6.1 Theory of Spatio-Temporal Filtering

Temporal-frequency content of a video is dependent on the spatial-frequency content of a key frame and the motion content of the video, which is discussed in Section 6.1.1. Hence, spatio-temporal filtering should not be viewed as arbitrary 3D filtering in space and time, but should be considered taking the motion content of video into account. To this effect, we discuss general principles of motion-adaptive filtering with motion detection in Section 6.1.2 and general principles of motion-compensated filtering with motion estimation in Section 6.1.3.

### 6.1.1 Frequency Spectrum of Video

We first define a motion trajectory and then derive the frequency spectrum of video for the case of global translational motion. Each pixel follows a curve in the $(x_1, x_2, t)$ space, called a "motion trajectory," which can be formally defined as a vector function $\mathbf{c}(t; x_1, x_2, t_0)$ that specifies the horizontal and vertical coordinates $(x_1', x_2')$ at time $t'$ of a reference pixel $(x_1, x_2)$ at time $t_0$, i.e., $(x_1', x_2') = [c_1(t'; x_1, x_2, t_0), c_2(t'; x_1, x_2, t_0)]$ [Dub 92]. The motion trajectory is illustrated in Figure 6.1. Given the motion trajectory $\mathbf{c}(t; x_1, x_2, t_0)$, the velocity of a pixel $(x_1', x_2')$ along the trajectory at time $t'$ can be defined by

$$\mathbf{v}(x_1', x_2', t') = \frac{d\mathbf{c}}{dt}(t; x_1, x_2, t_0)\Big|_{t=t'}$$

The fundamental assumption in motion estimation/compensation is that the intensity of a pixel remains unchanged along a motion trajectory, which places a

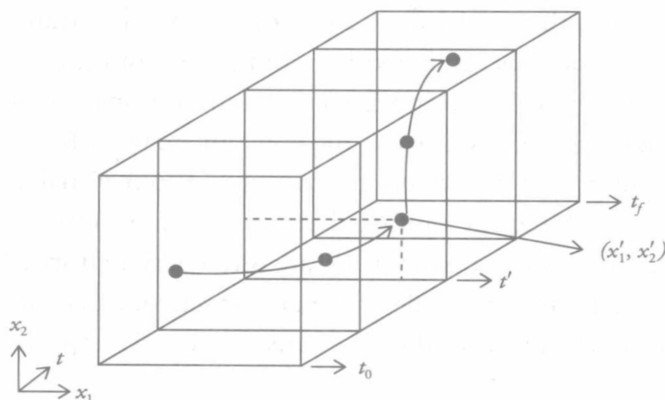



**Figure 6.1** Motion trajectory passing through pixel $(x_1', x_2')$ at time $t'$.

constraint on the local spatio-temporal spectrum of video. More specifically, local temporal-frequency content of video depends on local spatial-frequency content and the motion. We illustrate this concept for constant-velocity translation motion below.

### Constant-Velocity Global Translation

A simple model of image-plane motion is a global translation with constant velocity $(v_1, v_2)$, where frame-to-frame intensity variations can be modeled as

$$s_c(x_1, x_2, t) = s_c(x_1 - v_1 t, x_2 - v_2 t, 0) \equiv s_0(x_1 - v_1 t, x_2 - v_2 t) \tag{6.1}$$

where the reference (key) frame is chosen as $t_0=0$ and $s_0(x_1, x_2)$ denotes the 2D-intensity function of the reference frame.

In order to derive the spatio-temporal spectrum of video with constant-velocity global motion, we first define the Fourier transform of an arbitrary spatio-temporal function as (see Chapter 1)

$$S_c(F_1, F_2, F_t) = \iiint s_c(x_1, x_2, t)\, e^{-j2\pi(F_1 x_1 + F_2 x_2 + F_t t)}\, dx_1\, dx_2\, dt \tag{6.2}$$

where the inverse Fourier transform relationship is given by

$$s_c(x_1, x_2, t) = \iiint S_c(F_1, F_2, F_t)\, e^{j2\pi(F_1 x_1 + F_2 x_2 + F_t t)}\, dF_1\, dF_2\, dF_t$$

The support of $S_c(F_1, F_2, F_t)$ may occupy the entire $(F_1, F_2, F_t)$ space for arbitrary intensity functions $s_c(x_1, x_2, t)$. Next, we substitute (6.1) into (6.2) to obtain

$$S_c(F_1, F_2, F_t) = \iiint s_0(x_1 - v_1 t, x_2 - v_2 t) e^{-j2\pi(F_1 x_1 + F_2 x_2 + F_t t)}\, dx_1\, dx_2\, dt$$

Making the change of variables $x_i' = x_i - v_i t$, for $i = 1, 2$, we get

$$S_c(F_1, F_2, F_t) = \iint s_0(x_1', x_2') e^{-j2\pi(F_1 x_1' + F_2 x_2')}\, dx_1'\, dx_2' \int e^{-j2\pi(F_1 v_1 + F_2 v_2 + F_t)t}\, dt$$

which can be simplified as

$$S_c(F_1, F_2, F_t) = S_0(F_1, F_2)\, \delta(F_1 v_1 + F_2 v_2 + F_t) \tag{6.3}$$

where $S_0(F_1, F_2)$ is the 2D Fourier transform of $s_0(x_1, x_2)$ and $\delta(\cdot)$ is the 1D Dirac delta function. Defining $\mathbf{F} = [F_1\ F_2\ F_t]^{\mathrm{T}}$ and $\mathbf{v} = [v_1\ v_2\ 1]^{\mathrm{T}}$, the delta function in Eqn. (6.3) is

non-zero only when its argument $\mathbf{F}^T\mathbf{v}=\mathbf{0}$. The delta function thus confines the support of $S_c(F_1, F_2, F_t)$ to a plane in $R^3$ given by

$$F_1 v_1 + F_2 v_2 + F_t = 0$$

which passes through the origin, and is orthogonal to the vector $\mathbf{v}$. The spectral support of video with uniform velocity global motion is depicted in Figure 6.2(a).

This result is intuitively clear. Since the reference frame is sufficient to determine all future frames by the nature of the motion model, we expect that the Fourier transform of the reference frame would be sufficient to represent the entire spatio-temporal-frequency spectrum of video.

The extent of the support of the continuous video spectrum $S_c(F_1, F_2, F_t)$ on the plane $\mathbf{F}^T\mathbf{v}=0$ is determined by the support of $S_0(F_1, F_2)$. We assume that $s_0(x_1, x_2)$ is bandlimited, i.e., $S_0(F_1, F_2)=0$ for $|F_1|>B_1$ and $|F_2|>B_2$, then clearly, $s_c(x_1, x_2, t)$ is also bandlimited in the temporal variable, i.e., $S_c(F_1, F_2, F_t)=0$ for $|F_t|>B_t$, where

$$B_t = B_1 v_1 + B_2 v_2$$

For simplicity of illustration, the projection of the support of $S_c(F_1, F_2, F_t)$ into the $(F_1, F_t)$ plane, defined by $F_1 v_1 + F_t = 0$, is depicted in Figure 6.2(b).

Even if the global motion assumption is not valid, this motion model is commonly used for local video blocks in many applications, including international video compression standards. Hence, the spectrum model derived here is often

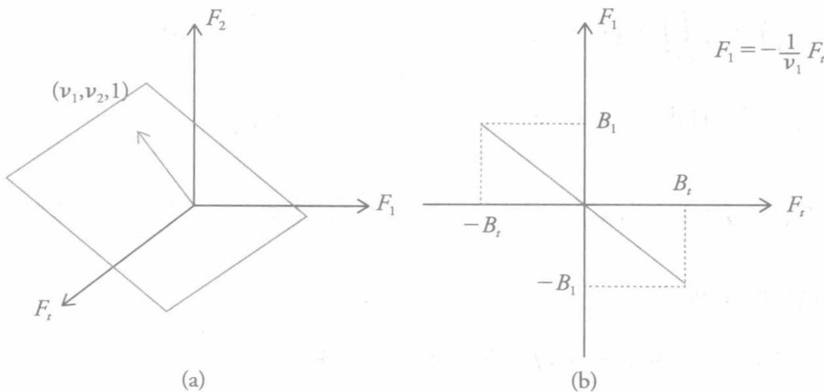



**Figure 6.2** Spectral support of video with uniform velocity global motion: (a) in the $(F_1, F_2, F_t)$ space and (b) projection of the spectral support onto the $(F_1, F_t)$ plane.

applicable to short-term (windowed) Fourier transform of local video blocks. If the motion within a block deviates from uniform translation, then the spectrum is still concentrated around a plane determined by the best translation vector.

### 6.1.2 Motion-Adaptive Filtering

Motion-adaptive filtering refers to employing different filtering strategies in the presence and absence of motion without the need for motion estimation. Motion-adaptive filters may employ explicit or implicit motion adaptation. Explicit schemes employ motion detection, and the form of the filter depends on the value of the motion-detection function. Several motion-detection methods, ranging from simple frame difference to more sophisticated temporal-integration schemes, for progressive video frames have been discussed in Section 5.2.2. For interlaced video input, we employ field differences, between the same parity fields, for motion detection. In implicit motion-adaptive filtering, there is no explicit motion detection, and motion adaptivity is inherent in the filter structure, such as the case in median filtering. We elaborate on some commonly used motion adaptive de-interlacing and frame-rate conversion algorithms in Sections 6.2.2 and 6.2.3, respectively.

### 6.1.3 Motion-Compensated Filtering

Motion-compensated (MC) filtering refers to filtering along motion trajectories at each pixel of each frame. Although MC filtering can be defined for arbitrary motion trajectories, it is the optimum linear shift-invariant filter for video with constant-velocity global motion. This is because the support of the spatio-temporal frequency response of a properly MC linear shift-invariant filter matches the support of the spatio-temporal frequency spectrum of video with constant-velocity global motion.

#### Arbitrary-Motion Trajectories

We define MC filtering along an arbitrary-motion trajectory by [Dub 92]

$$y(x_1, x_2, t) = \boldsymbol{F}\left\{s_c\left(c_1(\tau; x_1, x_2, t), c_2(\tau; x_1, x_2, t), \tau\right)\right\} \tag{6.4}$$

where $\boldsymbol{F}$ is a 1D linear or non-linear filter along the motion trajectory passing through $(x_1, x_2, t)$. In the most general case, a different motion trajectory $\mathbf{c}(\tau; x_1, x_2, t)$ may be defined for each pixel $(x_1, x_2)$ of frame at $t$, and $\tau$ ranges over all frames within the temporal support of the filter. The resulting MC filter is shift-invariant only if all

motion trajectories at each pixel of each frame are parallel to each other. This is the case when we have constant-velocity global motion. Motion-compensated filtering in the case of global but accelerated (in time) motion, or in the case of any space-varying motion, will result in a shift-varying filter. Obviously, a spatio-temporal frequency domain analysis of MC filtering can only be performed in the linear shift-invariant case, which is discussed next.

**Linear Shift-Invariant Filtering in the Case of Constant-Velocity Global Motion**

Given the velocity (motion vector) estimate $(v_1, v_2)$, the impulse response of an MC spatio-temporal filter can be expressed as

$$h(x_1, x_2, t) = h_1(t)\,\delta(x_1 - v_1 t, x_2 - v_2 t) \tag{6.5}$$

where $h_1(t)$ is the impulse response of the 1D filter applied along the motion trajectory.

Hence, linear shift-invariant filtering operation along a constant-velocity motion trajectory can be expressed as

$$\begin{aligned} y(x_1, x_2, t) &= \iiint h_1(\tau)\delta(z_1 - v_1\tau, z_2 - v_2\tau)\, s_c(x_1 - z_1, x_2 - z_2, t - \tau)\, dz_1\, dz_2\, d\tau \\ &= \int h_1(\tau) s_c(x_1 - v_1\tau, x_2 - v_2\tau, t - \tau)\, d\tau \end{aligned}$$

The frequency response of the MC filter can be found by taking the 3D Fourier transform of the impulse response (6.5)

$$\begin{aligned} H(F_1, F_2, F_t) &= \iiint h_1(t)\delta(x_1 - v_1 t, x_2 - v_2 t) e^{-j2\pi(F_1 x_1 + F_2 x_2 + F_t t)}\, dx_1\, dx_2\, dt \\ &= \int h_1(t) e^{-j2\pi(F_1 v_1 + F_2 v_2 + F_t)t}\, dt \\ &= H_1(F_1 v_1 + F_2 v_2 + F_t) \end{aligned} \tag{6.6}$$

Observe that the frequency response of the filter $H(F_1, F_2, F_t)$ is constant on planes $F_1 v_1 + F_2 v_2 + F_t = F$. If the spectrum of the input is confined to the plane $F_1 v_1 + F_2 v_2 + F_t = 0$, then the effect of filtering is simply multiplication by $H_1(0)$.

In general, the spectrum of input video extends out of the plane due to deviation of motion from constant-velocity translation and the presence of occlusion and noise. Then, a low-pass filter $H_1(F)$ is employed to attenuate frequencies out of the plane,

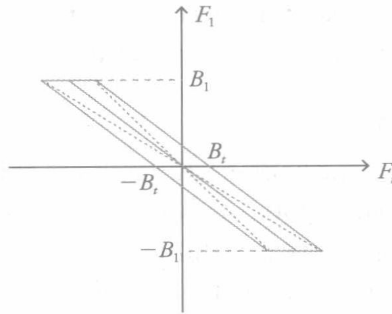



**Figure 6.3** Frequency response of the MC filter projected onto the $(F_1, F_t)$ plane. The passband has width $2B_t$ and slope $-v_1$. The solid line is the spectrum of input with matching velocity. The dotted lines indicate the tolerance range $\Delta$.

which results in a spatio-temporal low-pass filter $H(F_1, F_2, F_t)$ with a finite passband around the plane. The passband of the 3D spatio-temporal filter projected onto the $(F_1, F_t)$ plane is a parallelogram. The projected ideal filter $H_p(F_1, F_t)$ is given by

$$H_p(F_1, F_t) = \begin{cases} 1 & -B_1 < F_1 < B_1 \text{ and } -v_1 F_1 - B_t < F_t < -v_1 F_1 + B_t \\ 0 & \text{otherwise} \end{cases} \tag{6.7}$$

whose support is depicted in Figure 6.3. Proper motion compensation is achieved when $v_1$ matches the velocity of the input video. The case $v_1 = 0$ corresponds to pure temporal filtering with no motion compensation.

Because the ideal filter is unrealizable, it needs to be approximated, usually by an finite impulse response (FIR) filter, which poses a trade-off between filter length and passband width. In practice, the number of frame stores is limited, which necessitates the use of short temporal filters. Typically zero-order hold or pixel averaging along the motion trajectory is used. This causes a wider than desired transition band in the temporal-frequency dimension, which limits the aliasing or noise-rejection capability of the filter in resampling and de-noising applications, respectively. A number of other filter design issues for MC filtering are addressed in [Gir 93].

### Sensitivity to Errors in Motion Estimation

Because the MC low-pass filter has a finite passband width $B_t$, as shown in Figure 6.3, an MC filter based on an estimated velocity $v_{est}$ is capable of successfully filtering video with actual velocity within a range $[v_{est} - \Delta, v_{est} + \Delta]$, where the tolerance $\Delta$ depends on $B_t$ [Gir 85]. A filter with a wider passband yields a larger tolerance to motion-estimation errors, but may also pass spectral replications or noise within

passband range. A smaller passband means better suppression of spectral replications or noise, but smaller tolerance to motion-estimation errors.

### Reliable Motion Estimation

MC filtering has become practical after advances in hardware and software motion estimation solutions in the late 1990s. Various motion-estimation methods, including forward and backward block-matching, phase-correlation, and optical-flow methods, were discussed in Chapter 4. In MC filtering, estimation of true motion is preferred over finding the local minimum of a criterion function. Hence, several motion estimation schemes have been optimized for particular MC filtering applications [Bie 86, Caf 90, Tub 93, Haa 93, Yam 94, Hei 11].

In particular, MC de-interlacing and frame-rate conversion requires estimation of motion trajectories that pass through missing pixel locations. *Symmetric block-matching* is an extension of block-matching that is developed for this purpose. It is illustrated in Figure 6.4, where blocks in two existing neighboring frames/fields $k-1$ and $k+1$ are moved symmetrically, so that the line connecting the centers of these two blocks always passes through the missing pixel of $(x_1, x_2)$ in frame $k$ to define the motion trajectory for the missing pixels [Tho 89].

The accuracy and consistency of the motion estimates is probably the most important factor in the effectiveness of MC filtering. Thus, some kind of post-processing is usually applied to the estimated motion vectors to improve their accuracy.

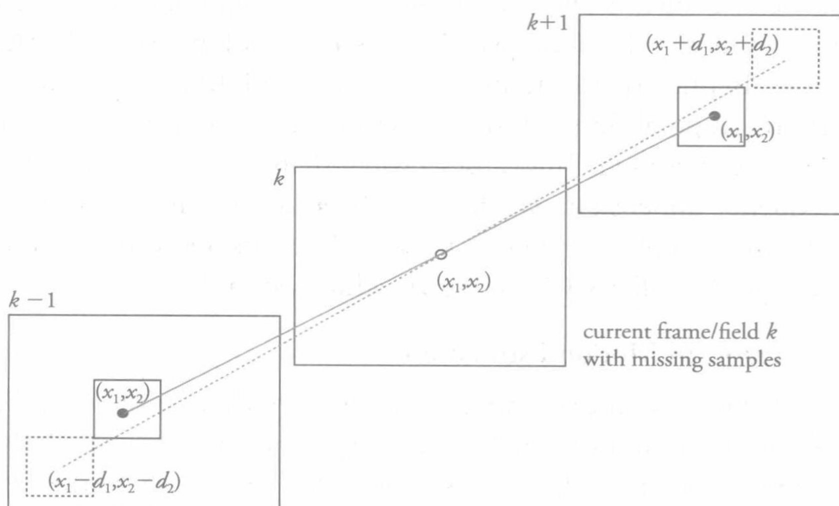



**Figure 6.4** Symmetric block-matching for MC up-sampling applications.

### Post-Processing of Motion Estimates and Occlusion Handling

The accuracy of the motion estimates is crucial for MC filtering; hence, post-processing for removing outlier motion estimates and occlusion handling is necessary in order to decide whether to use MC filtering or go into a fallback mode of motion-adaptive or intra-filtering at all pixels. We present two alternative approaches to test the accuracy of the motion estimates: an occlusion-detection method and a displaced frame-difference (DFD) method. Note that these methods are applied to the two frames used in symmetrical block-matching.

Occlusion detection is based on the assumption that a corona of motion vectors is located around moving objects. This assumption, coupled with the observation that the correct motion vectors from frame $k$ to $k+1$ map into "changed region" $CD(k, k+1)$, leads to the following procedure: If an estimated motion vector from frame $k$ to $k+1$ maps to the outside of the changed region, it indicates a pixel in frame $k$ that will be covered in frame $k+1$. Likewise, if a motion vector from frame $k+1$ to $k$ maps to outside of the changed region, it indicates a pixel in frame $k+1$ that is uncovered. Such motion vectors are unreliable and should be discarded.

An alternative method to detecting unreliable motion vectors is to test the DFD between the frames $k$ and $k+1$, where all vectors yielding a DFD above a pre-specified threshold are discarded. The unreliable motion vectors can be replaced by a set of candidate motion vectors if any of them yields a DFD that is less than the threshold. The candidate vectors can be determined based on the analysis of the histogram of the reliable motion vectors [Lag 92]. If no reliable replacement motion vector can be found at a pixel, it is marked as a motion-estimation failure, where a fallback-mode motion-adaptive or an intra-frame filter is employed.

## 6.2 Video-Format Conversion

A video format consists of a spatial resolution (picture size) and a frame rate (frequency) as discussed in Chapter 2. In the era of digital multimedia, various progressive and interlaced video formats are used to capture, store, transmit, and display digital video, including those for SD/HD TV broadcast, digital cinema, web media, phones, and camcorders. Format conversion is required to ensure interoperability of various applications by decoupling the spatio-temporal resolution requirements of the source from that of the display. The task of converting digital video from one format to another is referred to as video-format (standards) conversion, which includes interlacing/de-interlacing and frame/field rate down-/up-conversion. Both frame/field rate conversion and de-interlacing are based on the same principle of

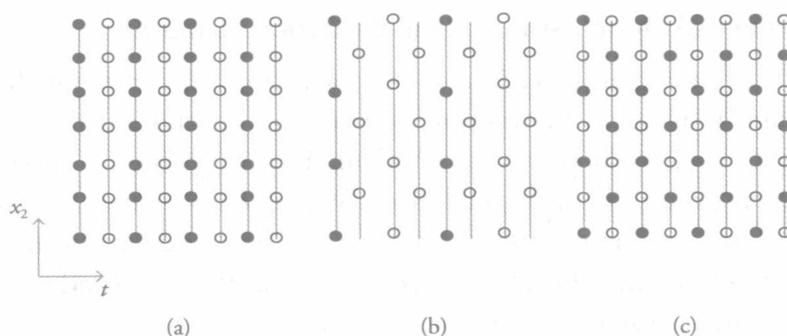



**Figure 6.5** Video-format conversion problems: (a) frame-rate up-conversion; (b) field rate (scan) up-conversion; and (c) de-interlacing. In all figures, filled circles show existing image lines and open circles indicate missing lines to be interpolated.

sampling-structure conversion; the only difference between them is the structure of the input and output lattices. Video-format conversion is necessary in such applications as displaying 50i/60i broadcast video in 100/120 Hz interlaced or progressive displays to reduce flicker, exchanging broadcast video content between 50 Hz and 60 Hz countries, converting digital cinema at 24 fps to 50/60 Hz TV broadcast formats, and in post-production workflow when combining content shot at different frame rates or inserting overlays and special effects.

Most of the common video-format conversion problems are illustrated in Figure 6.5. Frame and field rate up-conversion increase the temporal sampling rate in progressive and interlaced video, respectively. De-interlacing refers to up-conversion from interlaced to progressive video. In addition to enabling reuse of video captured at different frame rates, frame-rate up-conversion generally yields higher visual quality through better motion rendition and less flicker, and de-interlacing provides improved spatial resolution in the vertical direction.

All video-format conversion problems deal with sampling structure conversion, which was first introduced in Section 1.5, where it was discussed for arbitrary input and output sampling structures and arbitrary signals within the framework of linear filtering. Here, we extend that framework by incorporating motion models that characterize temporal variations in video. We first treat down-conversion in Section 6.2.1, where we show that unlike the case of still images, in some cases sub-sampling without anti-alias filtering may be desirable in video processing, since aliasing can be used to recover some frequencies beyond the Nyquist frequency for super-resolution reconstruction. It is clear from our discussion in Section 6.1 that video-format conversion requires designing truly spatio-temporal filters, taking into consideration the spatio-temporal bandwidth of video before and after the conversion. However, in

many practical applications, spatial and temporal filtering are considered separately for ease of design and implementation. When performing MC frame-rate conversion starting with an interlaced source, de-interlacing is often employed first for correct motion rendering, even if the desired output format is also interlaced. For MC filtering, often a spatio-temporal filter in the direction of motion is used. We introduce some commonly used filters for de-interlacing and frame-rate conversion in Section 6.2.2 and Section 6.2.3, respectively.

## 6.2.1 Down-Conversion

Recall from Section 1.5 that down-conversion refers to anti-alias filtering (optional) followed by down-sampling (sub-sampling). Suppose we down-sample from an MD input lattice $\Lambda_1$ with the sampling matrix $\mathbf{V}_1$ to an MD output lattice $\Lambda_2$ with the sampling matrix $\mathbf{V}_2$. We define an $M \times M$ sub-sampling matrix $\mathbf{S} = \mathbf{V}_1^{-1}\mathbf{V}_2$ such that $\mathbf{V}_2 = \mathbf{V}_1\mathbf{S}$. Since $\mathbf{V}_1$ is always invertible, a sub-sampling matrix can always be defined. Then, the sites of the output lattice $\Lambda_2$ can be expressed as

$$\mathbf{y} = \mathbf{V}_1 \mathbf{S} \mathbf{n}, \quad \mathbf{n} \in \mathbf{Z}^M \tag{6.8}$$

For example, spatio-temporal sub-sampling according to the matrix

$$\mathbf{S} = \begin{bmatrix} 2 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

corresponds to 2:1 interlacing, i.e., discarding even and odd columns in alternating frames, respectively. Down-conversion with or without anti-alias filtering is discussed next.

### Down-Conversion with Anti-Alias Filtering

Should the Fourier spectrum of input video extend to outside the unit cell of the reciprocal lattice $\mathbf{V}_2^*$, an anti-alias filter is needed before down-sampling in order to avoid aliasing. The anti-alias filter attenuates all frequencies within the reciprocal lattice $\mathbf{V}_1^*$ that fall outside of the unit cell of the reciprocal lattice $\mathbf{V}_2^* = (\mathbf{V}_1\mathbf{S})^*$ so that the spectral replications centered about the sites of the reciprocal lattice $\mathbf{V}_2^*$ do not overlap with each other. However, because of this anti-alias filtering, the high frequencies in the input video are permanently lost, and a viewer whose eyes track a moving object can perceive spatial blurring. Furthermore, if the video needs to be up-converted for display purposes, the original spectral content of the video cannot be recovered even by using ideal MC-reconstruction filters. This is illustrated in the

"Comparison of Down-Conversion with or without Anti-aliasing" example below. Clearly, in this case, successive frames contain no new frequency information, and a simple linear shift-invariant low-pass filter is as good as a more sophisticated MC-reconstruction filter.

### Example: Conversion from ITU-R 601 4:2:2 to ITU-R 601 4:2:0 Format

Conversion from 4:2:2 interlace studio format to 4:2:0 interlace for compression and broadcast requires decimation of both chroma channels vertically by a factor of 2. This is complicated by the interlaced nature of the source, since simple vertical decimation by 2 of each field would not preserve the spatial offset between the lines of two fields. It can be treated by filtering one of the chroma fields (in the vertical direction) using a filter of odd length before eliminating every other line and processing the alternate chroma field by using an even length filter to offset the locations of lines by 1/2 sample. The odd- and even-length filter impulse responses are [-29, 0, 88, 138, 88, 0, -29]/256, and [1, 7, 7, 1]/16, respectively. Note that the luma channel is not modified in this conversion.

### Down-Conversion without Anti-Alias Filtering

Down-conversion without anti-alias filtering refers to simply discarding a subset of the input samples specified by a sub-sampling matrix $\mathbf{S}$. Unlike the case of still images, this process can preserve the high-frequency content of the original video except for the case of critical velocities, provided that we have global, constant-velocity motion. Even if the video technically contains aliasing, a viewer whose eyes track the moving object does not see aliasing artifacts and perceive the moving object with the original spatial detail. Furthermore, MC linear shift-invariant filtering can be employed for subsequent up-conversion of video without any loss of spatial resolution. This is demonstrated by the following example.

### Example: Comparison of Down-Conversion with or without Anti-aliasing

For the sake of simplicity, let's consider a single spatial (horizontal) coordinate, $x$, and a time coordinate, $t$, and assume we have horizontal motion with constant velocity $v$. Suppose that the output lattice $\mathbf{V}_2$ is obtained by sub-sampling a progressive input lattice $\mathbf{V}_1$ according to the sub-sampling matrix

$$\mathbf{S} = \begin{bmatrix} 2 & 1 \\ 0 & 1 \end{bmatrix}$$

which corresponds to discarding even and odd spatial samples in alternating time samples. The spatio-temporal $(x, t)$ input and output lattices in this case are depicted in Figure 6.6.

The spectrum of the input video, assuming global, constant-velocity motion, is depicted in Figure 6.7(a), where the solid dots indicate the sites of the reciprocal lattice $\mathbf{V}_1^*$ and the slope of lines are determined by the velocity $v$. The dotted lines denote the support of an ideal anti-alias filter. The spectra of the down-converted signal with and without anti-alias filtering are shown in

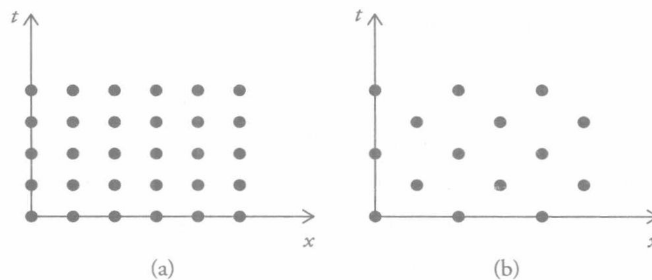



**Figure 6.6** Two-dimensional (a) input and (b) output lattices.

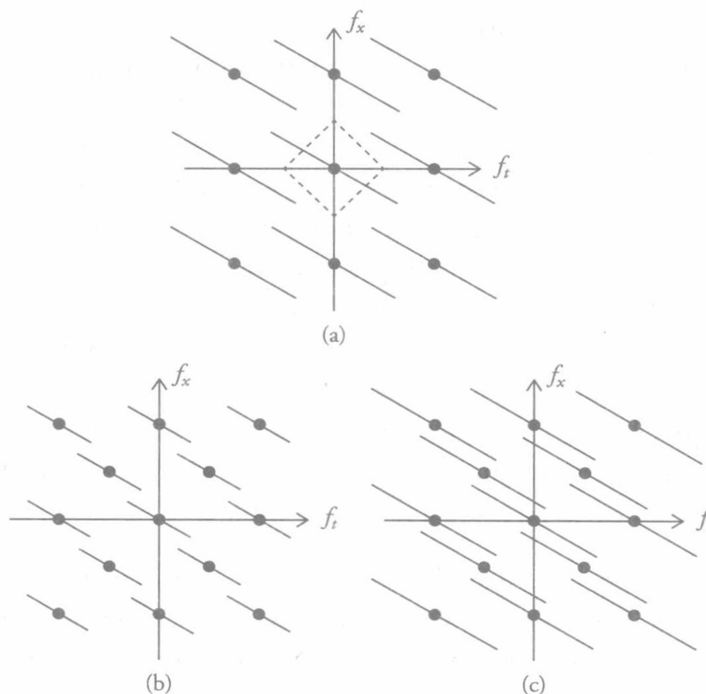



**Figure 6.7** Fourier spectrum of (a) input signal; (b) sub-sampled signal with anti-alias filtering; and (c) sub-sampled signal without anti-alias filtering.

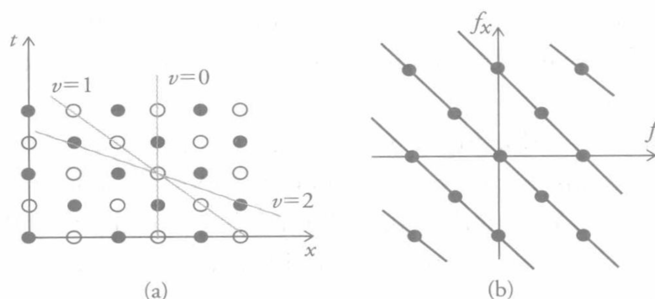



**Figure 6.8** Critical velocities in the case of de-interlacing: (a) the spatio-temporal domain and (b) the Fourier domain.

Figures 6.7(b) and (c), respectively. The loss of high-frequency information when an anti-alias filter has been employed can be clearly observed. Further observe that the spectrum of the down-converted signal in Figure 6.7(c) retains the high-frequency content of the original signal, provided that we do not have a critical velocity, discussed below.

If the velocity $v$ is such that the orientation of the lines (spectra) aligns with sites of the reciprocal lattice, it is called a critical velocity. Then, the replications overlap with each other, resulting in aliasing (loss of information) as shown in Figure 6.8(a). Given the sub-sampling matrix, we can easily determine the critical velocities. For the sub-sampling matrix in the "Comparison of Down-Conversion with or without Anti-aliasing" example, the velocities $v_c = 2i+1$, $i \in Z$, are critical velocities. For these velocities, the motion trajectory passes through either all existing or non-existing pixel sites at every frame in spatio-temporal domain, as shown in Figure 6.8(b). Note that with proper anti-alias filtering, the replicas in the frequency domain can never overlap; hence, there are no critical velocities.

In conclusion, down-conversion to sub-Nyquist rates followed by up-conversion without loss of resolution is possible only if no anti-alias filtering has been used in the down-conversion, and we do not have a critical velocity. If the estimated velocity vector at a particular frame or a block is close to this critical velocity, we need to choose one of the following options before sub-sampling:

1. Anti-alias filtering: The spatial resolution is sacrificed at a particular frame or block of pixels when the estimated motion is close to the critical velocity.
2. Adaptive sub-sampling: If we are allowed to change the sub-sampling lattice, given an estimate of the motion vector, we may change the sub-sampling matrix to avoid critical velocities assuming global, constant-velocity motion.

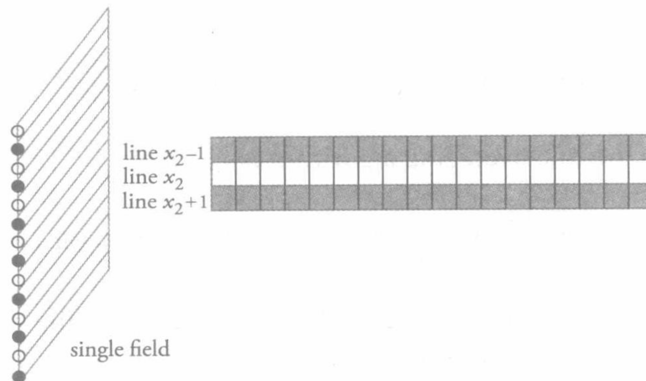



**Figure 6.9** Intra-field filtering.

## 6.2.2 De-Interlacing

The process of interlace-to-progressive scan conversion is called de-interlacing. De-interlacing techniques complete each field to a frame by estimating (interpolating) the missing lines, which are depicted by open circles in Figure 6.5(c). De-interlacing methods can be classified as intra-field vs. inter-field methods, where the latter can be further classified as motion-adaptive and MC methods.

### Intra-Field De-Interlacing

Intra-field de-interlacing refers to interpolating missing lines from the available lines within a single field. A field of an interlaced video is depicted in Figure 6.9, where each circle denotes the cross-section of a complete line of video. The filled circles denote lines that are available, and the open circles show lines to be interpolated. Intra-field de-interlacing methods include linear (vertical) or edge-adaptive interpolation filtering. Intra-field filters do not generate motion artifacts; however, they may cause aliasing or loss of vertical resolution mainly around horizontal edges.

#### *Linear Interpolation*

These methods are called "bob" filtering in the computer industry. Let $s(x_1, x_2, t_i)$, $i=e,o$, denote the even and odd fields, respectively, at time $t$, such that $s(x_1, x_2, t_e)$ is zero for odd values of $x_2$, and $s(x_1, x_2, t_o)$ is zero for even values of $x_2$.

***Line Repetition*** Assuming that the index of the first line of each field is zero, the line-repetition algorithm can be described by

$$s(x_1, x_2, t_e) = s(x_1, x_2 - 1, t_e) \quad \text{for } x_2 \text{ odd} \tag{6.9a}$$

and

$$s(x_1, x_2, t_o) = s(x_1, x_2+1, t_o) \quad \text{for } x_2 \text{ even} \tag{6.9b}$$

*Line Averaging* The line-averaging algorithm is given by

$$s(x_1, x_2, t_i) = \frac{1}{2}[s(x_1, x_2 - 1, t_i) + s(x_1, x_2 + 1, t_i)] \quad \text{for } i = e, o \tag{6.10}$$

The line-repetition algorithm may result in jagged edges, while the line-averaging algorithm may cause undesired blurring.

*Edge-Adaptive Intra-Field Interpolation*

Edge-adaptive interpolation methods have been proposed to avoid blurring near edges [Lim 90, Lee 94]. In the edge-adaptive approach, each line of video in a field/frame $t_0$ is locally modeled as a horizontally displaced version of the previous line in the same field/frame as

$$s\left(x_1 - d/2 + j, x_2 - 1, t_0\right) = s\left(x_1 + d/2 + j, x_2 + 1, t_0\right)$$

where $d$ denotes the local horizontal displacement between two consecutive even or odd lines. This model suggests a 1D displacement-compensation problem where $x_2$ takes the place of the time variable in the motion compensation problem. The displacement $d$ at each pixel can be estimated by using symmetric line-segment matching about $(x_1, x_2)$ in order to minimize the sum absolute difference (SAD) given by

$$\text{SAD}(d) = \sum_{j=-1}^{1} \left| s\left(x_1 - d/2 + j, x_2 - 1, t_0\right) - s\left(x_1 + d/2 + j, x_2 + 1, t_0\right) \right| \tag{6.11}$$

or through the pixel-flow relationship [Lim 90]

$$d/2 \frac{\partial s(x_1, x_2, t_0)}{\partial x_1} + \frac{\partial s(x_1, x_2, t_0)}{\partial x_2} = 0 \tag{6.12}$$

which is similar to the optical-flow equation for the case of vertical edge-displacement estimation. Then an edge-adaptive contour-interpolation filter can be defined as

$$s(x_1, x_2, t_i) = \frac{1}{2}\left[s\left(x_1 - d/2, x_2 - 1, t_i\right) + s\left(x_1 + d/2, x_2 + 1, t_i\right)\right] \tag{6.13}$$

for $i = e, o$, where $e$ and $o$ denote even and odd fields, respectively. This filter seeks those two pixels in the two neighboring lines that most likely belong to the same

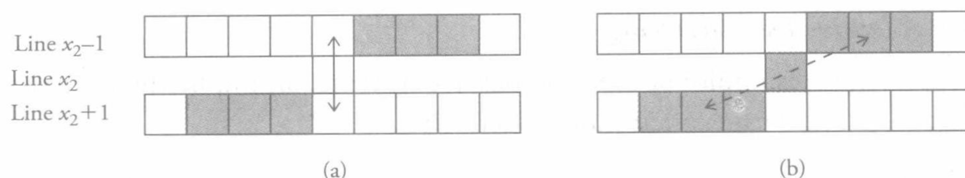



**Figure 6.10** Demonstration of (a) linear vs. (b) edge-adaptive vertical interpolation.

image structure, i.e., on the same side of the edge, and averages them. The fact that the filter is capable of preserving a 45-degree edge, unlike the linear averaging filter, is demonstrated in Figure 6.10. The crucial step here is the accurate estimation of local edge-displacement (orientation) values.

A hardware implementation for edge-adaptive line interpolation has been proposed [Lee 94]. Intra-frame filtering methods lead to simple hardware realizations. However, they are not well suited to de-interlacing in stationary regions, where spatial averaging usually causes blurring of image details, hence the need for motion-adaptive de-interlacing.

### Inter-Field Temporal De-Interlacing (Weave Filtering)

The simplest de-interlacing method is merging even and odd fields of a frame, i.e., copying samples as shown by the horizontal arrow in Figure 6.11(a), which yields $N/2$ progressive (composite) frames from $N$ fields, which are then replicated to obtain $N$ progressive frames. Composite frames provide perfect vertical resolution in stationary image regions, but they suffer from line-crawling artifacts in regions of fast motion. Field merging is called weave filtering in the computer industry.

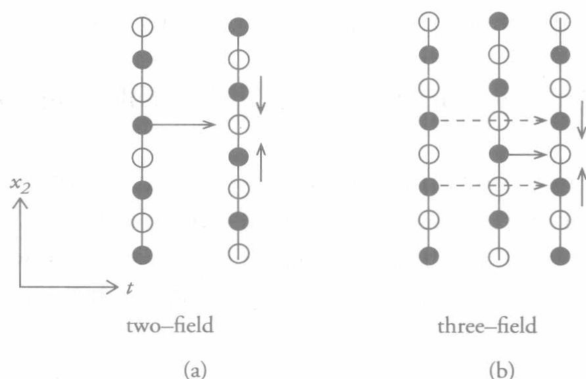



**Figure 6.11** Motion-adaptive de-interlacing: (a) two-field three-pixel median filter and (b) three-field filter with motion detection between the same parity fields.

### *Motion-Adaptive De-Interlacing*

We consider two examples of motion-adaptive filtering: an implicitly motion adaptive filter and one with explicit motion detection.

***Two-Field Median Filter*** An example of an implicit motion-adaptive filter that does not require motion detection is the three-point median filter, whose support is shown by the arrows depicted in Figure 6.11(a) [Cap 90, Haa 92]

$$s(x_1,x_2,t_i)=\text{Med}\left\{s\left(x_1-\frac{d}{2},x_2-1,t_i\right),s\left(x_1+\frac{d}{2},x_2+1,t_i\right),s\left(x_1,x_2,t_i-1\right)\right\} \quad (6.14)$$

It is popular due to its computational simplicity and its edge-preserving property.

***Bob-and-Weave Filter*** In order to obtain the best performance in both moving and stationary regions, we consider motion-adaptive filtering, which switches between weaving and intra-frame interpolation [Haa 98] or linearly blends them based on a motion-detection function [Sch 87]. A three-field bob-and-weave filter, whose support is depicted in Figure 6.11(b), is given by

$$s(x_1,x_2,t_i)=\alpha\, s\left(x_1-\frac{d}{2},x_2-1,t_i\right)+\beta\, s\left(x_1+\frac{d}{2},x_2+1,t_i\right)+\gamma\, s\left(x_1,x_2,t_i-1\right) \quad (6.15)$$

where parameters $\alpha$ and $\beta$ are determined based on the value of a motion-detection function,

$$\alpha = 0.5, \beta = 0.5, \gamma = 0 \text{ if motion is detected (Bob)}$$

$$\alpha = 0, \beta = 0, \gamma = 1 \text{ if motion is detected (Weave)}$$

and parameter $d$, given by Eqn. (6.12) or (6.13), enables edge-adaptive intra-frame interpolation.

We can employ three- or four-field motion detection. Three-field motion detection can be obtained by thresholding the difference between two fields of the same polarity (even-even or odd-odd) as depicted in Figure 6.11(b), whereas the four-field motion detection takes the logical OR of thresholded differences of the respective even-even and odd-odd fields.

Motion-adaptive methods provide satisfactory results, provided that the scene does not contain global camera motion or fast-moving objects. In the presence of camera motion or fast-moving objects, MC filtering is needed for better results.

## Motion-Compensated De-Interlacing

Motion compensation aims to transform a video with global or local motion into a stationary sequence. We investigate MC filtering under two cases: i) global MC filtering in the case of camera motion, and ii) general MC filtering in the case of fast-moving objects, which typically requires a different motion trajectory at each pixel.

### *Global-Motion-Compensated De-Interlacing*

In the case of a global camera motion or camera shake, a hybrid de-interlacing method can be employed, which first compensates for the dominant global motion, and then applies a motion-adaptive filter on this MC image to account for any residual motion [Pat 97a]. The block diagram of a three-field hybrid de-interlacing filter is depicted in Figure 6.12, where three consecutive fields are assumed to be an even field $E_1$, odd field $O_1$, and even field $E_2$.

In the first stage, a global-motion vector between the fields $E_1$ and $E_2$ is estimated using the phase-correlation method over four rectangular windows that are located near the borders of the fields, so they are most likely only affected by global motion. Next, the fields $O_1$ and $E_2$ are motion compensated with respect to $E_1$ to generate $O_1'$ and $E_2'$, respectively. The motion-compensation step aims to create three consecutive fields, $E_1$, $O_1'$, and $E_2'$, where the global motion is eliminated. Subsequently, the three-field motion-adaptive bob-and-weave filter is applied to the field sequence $E_1$, $O_1'$, and $E_2'$. A judder post-processing step, proposed by Zaccarin and Liu [Zac 93], can also be included [Pat 97a]. Judder refers to edge-misalignment artifacts, as shown in Figure 6.13, caused by incorrect motion vectors.

Judder detection may be implemented by detection of the staircase pattern in edge locations, which is shown in Figure 6.13(b). In the post-processing stage, motion vectors at the pixels where judder is detected are deemed unreliable, and the corresponding pixels are replaced by spatially interpolated values.

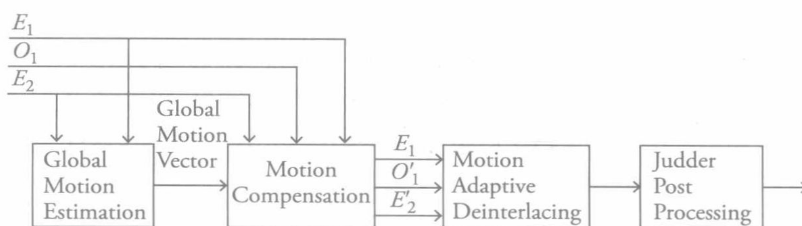



**Figure 6.12** Motion-compensated/adaptive de-interlacing.

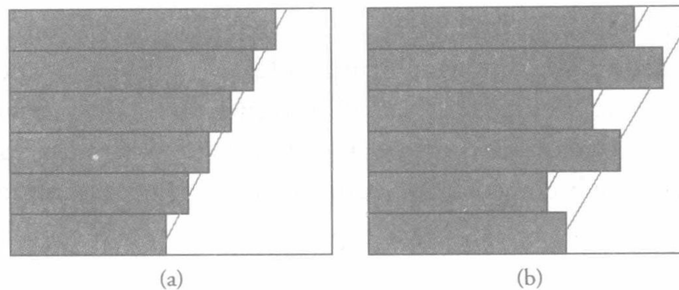



**Figure 6.13** Illustration of judder: (a) no judder and (b) judder (line crawling) present.

*General-Motion-Compensated De-Interlacing*

The basic concept of MC filtering is to perform filtering along motion trajectories passing through missing pixels. Motion-compensated up-conversion is capable of producing video with higher spatio-temporal resolution than the input, and free from aliasing, when the original source is spatio-temporally aliased, as discussed in Section 6.2.1, provided that i) the motion model is accurate at least on a local basis, ii) the motion estimates are accurate, and iii) we do not have a critical velocity. The performance of an MC filter is closely related to how well we deal with inaccurate motion vectors, occlusions, and critical velocities. The procedure consists of three steps: i) true (sub-pixel) motion estimation, ii) post-processing of motion vectors, and iii) choice of filter. We presented motion estimation and post-processing of motion vectors for MC filtering in Section 6.1.3. In the interest of real-time implementation with a small number of frame stores, often a simple zero-order hold (weave) filter is employed along motion trajectories. Here, we present some variations of MC zero-order hold filtering.

***Backward Extension of Motion Vectors*** In MC zero-order hold (field-insertion) filtering, spatial interpolation within the previous field is required to accomodate sub-pixel motion vectors. In an attempt to minimize the need for spatial interpolation, Woods and Han [Woo 91] extend the search to find a match with one of the existing samples over two previous fields as shown in Figure 6.14. The lines indicate possible motion vectors (MV) and the pixel to be copied for each MV. For MVs that do not point to the proximity of an existing pixel location in either of the previous two fields, the average of the two nearest existing pixels in the previous two fields is computed as the interpolated value.

***Time-Recursive De-Interlacing*** Time-recursive (TR) filters use previously de-interlaced frames (instead of input fields only) in MC de-interlacing [Wan 90]. Since

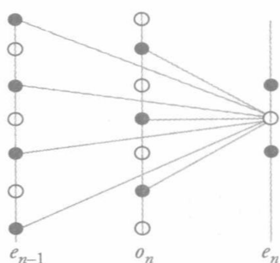



**Figure 6.14** Zero-order hold filtering extended to two previous fields.

interpolated samples depend on previous original samples as well as previously interpolated samples, TR filters may cause propagation of interpolation errors into subsequent frames. MC median filtering, rather than simple MC weaving, has been proposed to prevent error propagation.

***Hybrid Methods*** Successful and robust de-interlacing often requires adaptively combining MC and motion-adaptive methods, such as intra-line averaging (bob), edge-adaptive intra-averaging, field insertion (weave), and MC forward or backward field insertion and MC field averaging [Haa 98]. The fundamental difficulty with hybrid methods is to develop a systematic approach for reliable quality ranking of individual methods for proper switching.

### 6.2.3 Frame-Rate Conversion

Frame-rate conversion refers to presenting video at a different temporal rate than at which it was shot. The most common examples are displaying 50i/60i broadcast TV on 100/120 Hz interlaced or progressive panels and broadcasting 24 Hz movies on TV in a 50/60 Hz country. Frame-rate conversion techniques can range from simple frame replication (pull down) to very sophisticated motion compensated interpolation (synthesis) techniques.

#### 24 Hz Movies to 50/60 Hz

Movies are recorded at 24 frames/sec or 24 Hz. In most theatres, they are projected at 72 Hz via a shutter showing the same frame three times. Blu-ray disc specification requires 23.976 Hz 1920 × 1080 video. TV standards vary according to geographical location. The UK has a standardized TV frame rate of 50 Hz (fields/sec) or 25 interlaced frames/sec in the 1930s, while the TV frame rate in the United States is 60 (59.94) Hz or 30 (29.97) interlaced frames/sec. Pull-down methods are

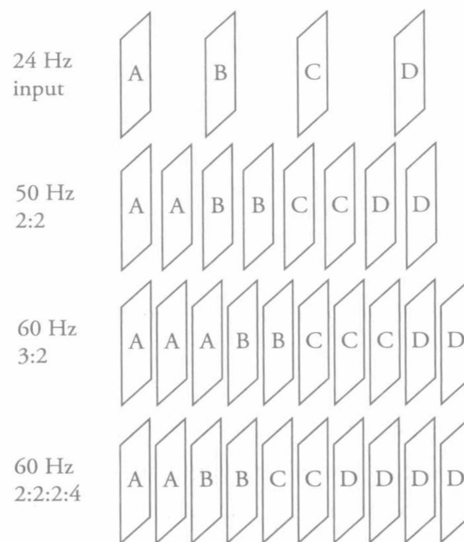



**Figure 6.15** Illustration of various pull-down methods.

frame-repetition methods that insert additional frames in various patterns so the duration of 24 Hz (23.976 Hz) movies is unchanged when shown on 50/60 Hz TV broadcasts.

In a 50 Hz European TV, 24 Hz movies are simply played back as if they are 25 pictures/sec by splitting each picture into two fields. This is called 2:2 pull-down, which is illustrated in Figure 6.15. The duration of the movie is reduced by 1/24 every second, but we do not notice this. Note that the audio pitch is shifted accordingly.

In the 60 Hz TV world speeding up a 24 Hz source to 60 Hz would be visible; therefore, 3:2 or 2:2:2:4 pull-down methods have been invented. In the 3:2 pull-down, each odd frame of digital motion picture is repeated three times and each even frame is repeated twice, or vice versa. These pictures are then interlaced yielding a 60 Hz field rate from a 24 Hz input source. Alternatively, 2:2:2:4 pull-down repeats the first three frames twice and the fourth frame four times periodically, before interlacing each picture. These repetition patterns are depicted in Figure 6.15. The pull-down methods introduce temporal aliasing that results in jerky motion rendition. Such motion artifacts may be visible with bigger displays and high-resolution video formats, where more sophisticated motion-adaptive or MC frame/field-rate conversion algorithms may be needed.

The inverse pull-down methods can be used to recover a 24 Hz progressive source from a 60 fields/sec interlaced video that was generated from motion pictures using 3:2 or 2:2:2:4 pull-down methods. The inverse pull-down methods have been

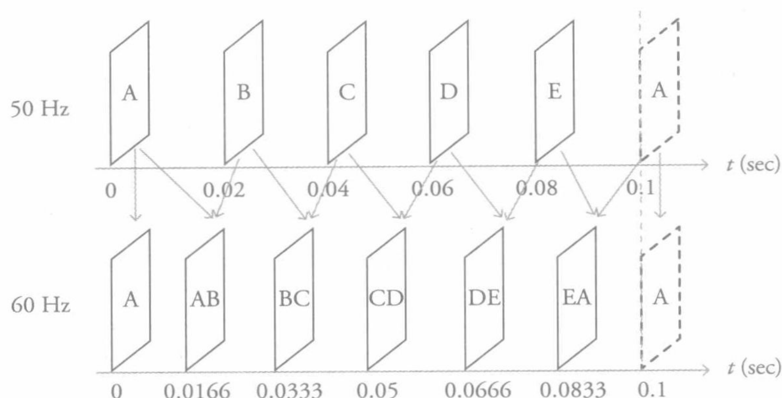



**Figure 6.16** 50 to 60 Hz conversion by weighted averaging (blending) of frames.

adopted by the Moving Picture Experts Group (MPEG) as a pre-processing step for redundancy reduction in compression of 60 Hz interlace video that was generated from motion pictures using the 3:2 pull-down method.

### 50 to 60 Hz Conversion

Conversion from 50 Hz to 60 Hz requires replicating a picture (an even and an odd field) every five pictures. Conversion from 60 Hz to 50 Hz may be achieved by dropping a picture every six frames. Smoother results can be obtained by weighted averaging interpolation (blending) in time (shown in Figure 6.16), rather than frame dropping or replicating, to achieve the desired field rate.

### Scan-Rate Doubling

Doubling of the field rate (also called the scan rate) has been adopted by most TV manufacturers of 100 Hz receivers to improve visual quality. In digital TV receivers, it is easy to replicate each field twice to achieve scan-rate doubling. There exists more than one way to repeat the fields. For example, an odd field may be repeated to form the next even field, and an even field is repeated to form the next odd field. This method has reasonably good performance in moving scenes, but poor results in stationary regions are inevitable. Alternatively, one can repeat an even field to form the next even field, and an odd field to form the next odd field. This strategy is optimal for stationary scenes but fails in moving regions.

A linear shift-invariant inter-frame filtering strategy for frame/field rate up-conversion would be frame/field averaging, where a missing frame is replaced by the average of its two neighboring frames/field. Averaging improves the SNR in stationary regions; hence, it is superior to simple frame/field repetition. However, it may introduce ghost artifacts in moving regions.

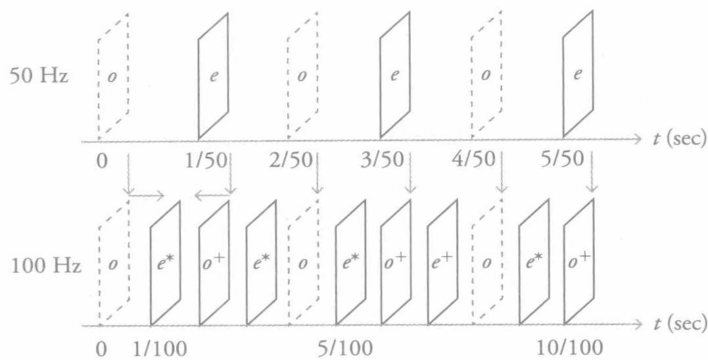



**Figure 6.17** Interlace to interlace scan-rate conversion.

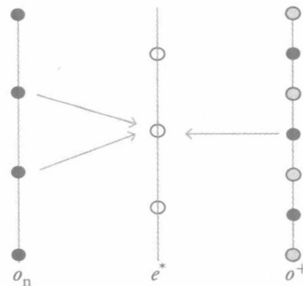



**Figure 6.18** Dark circles indicate lines existing in the input odd and even fields. Even fields $e^*$ are interpolated by the three-point filter. Light-color circles indicate output odd field lines interpolated from the input even field lines by intra-frame line averaging. Even lines indicated by dark circles in the field $o^+$ are discarded.

A simple scan-rate conversion method that preserves temporal sequencing of the fields is sequential intra- and inter-field line averaging, which is illustrated in Figure 6.17 and Figure 6.18. Only input odd fields (marked $o$) are retained "as is" in the 100 Hz output sequence. The input even fields are converted to odd fields (marked $o^+$) in the output video by intra-field line averaging. The interpolated odd field lines are shown by lighter circles in Figure 6.18. New even fields (marked $e^*$) are interpolated between $o$ and $o^+$ by inter-field filtering (blending). A three-point average or median filter, shown by the arrows in Figure 6.18, is used for interpolation of even fields. The three-point filter uses existing lines in the input odd and even fields.

Intra-field line-averaging performs reasonably well in moving regions. However, it introduces blurring in stationary regions because it is a spatial filter. Notice that field repetition and inter-field averaging are optimal (in noise-free and noisy

cases, respectively) in stationary regions. However, while field-repetition algorithms yield jagged edges, inter-field averaging may introduce ghost artifacts in moving regions. Obviously, non-adaptive, non-MC algorithms cannot be perfect for both stationary and moving regions, which suggests the need for motion-adaptive or MC filtering.

### Motion-Adaptive Scan-Rate Conversion

One way to avoid blurring in stationary regions and ghosting in moving regions is to employ an adaptive filter where the filter impulse response is determined locally based on a motion-detection function. For example, we can replicate the same parity fields in stationary regions and perform three-point weighted averaging in moving regions. The boundaries of the moving regions can be estimated by using a motion-detection function, which may simply be the frame difference as in change detection. Because no optimal strategy exists to determine the filter weights in terms of the motion-detection function in moving areas, several researchers have suggested the use of spatio-temporal median filtering. Median filtering is known to be edge-preserving in still-frame image processing. Considering the effect of motion as a temporal edge, spatio-temporal median filtering should provide motion adaptivity.

In scan-rate up-conversion, we employ a three-point spatio-temporal median filter described by

$$s(x_1, x_2, t_{e*}) = \text{Med}\left\{s(x_1, x_2 - 1, t_o), s(x_1, x_2 + 1, t_o), s(x_1, x_2, t_{o^+})\right\} \tag{6.16}$$

where "Med" denotes the median operation. The three pixels within the support of the filter are shown in Figure 6.18 for even-field estimation. The median filter is motion-adaptive such that in moving regions the filter output generally comes from one of the two pixels that is 1/2 pixel above or below the pixel to be estimated, and in the stationary regions it comes from the pixel that is in the same vertical position. Thus, it yields reasonable performance in both stationary and moving regions. The three-point median filter has been used in improved-definition TV receivers for field-rate doubling. Several modifications including a combination of averaging and median filters have also been proposed for performance improvements [Haa 92].

In median filtering, each sample in the filter support is given an equal emphasis. Alternatively, one can employ weighted median filtering, given by

$$s(x_1, x_2, t_{e*}) = \text{Med}\left\{w_1 \Diamond s(x_1, x_2 - 1, t_o), w_2 \Diamond s(x_1, x_2 + 1, t_o), w_3 \Diamond s(x_1, x_2, t_{o^+})\right\}$$

where $w_i$ denotes the weight of the $i$th sample and $\Diamond$ is the replication operator. That is, each sample $i$ is replicated $w_i$ times to affect the output of the median operation. To obtain the best possible results, the weights should be chosen as a function of a motion-detection signal. An example of a motion-detection signal for scan-rate up-conversion and weight combination as a function of the motion-detection signal is given by Haavisto and Neuvo [Haa 92].

**Motion-Compensated Frame/Scan-Rate Conversion**

If frame/field duplication/deletion or simple linear averaging (blending) of frames or median filtering do not produce satisfactory results, new frames need to be synthesized by MC interpolation. There is usually a trade-off between the accuracy of spatial-image details (reusing an original frame) and spatio-temporal accuracy (positions of objects in a frame at a given time according to motion trajectories). If we reuse an existing frame at a slightly different time than it actually belongs to, spatial details will be well-preserved but positional inaccuracy may lead to motion jitter. If we synthesize a new frame for the exact time position then spatial position of objects will be correct (assuming true motion estimation) but some spatial details may be blurred due to interpolation errors. A simple method for frame synthesis is MC blending (i.e., averaging of pixels from neighbor frames along the motion trajectory passing through the center pixel). Low-cost digital frame buffers and real-time motion estimation hardware/software have made MC filtering practical since the late 1990s. Commercial and non-commercial implementations of these algorithms are available with good results. ASIC implementations are built into high frame-rate TVs under different brand names.

An MC frame/field rate conversion system consists of several sub-blocks [Bar 10], which are shown in Figure 6.19. We note that if the source video is interlaced, de-interlacing is applied prior to MC filtering for more accurate motion estimation and frame rendering even if the desired output is interlaced. Motion-estimation and post-processing methods were discussed in Section 6.1.3 and de-interlacing methods were covered in Section 6.2.2. Frame/field replication or three-point median filtering is often used as a fallback mode.

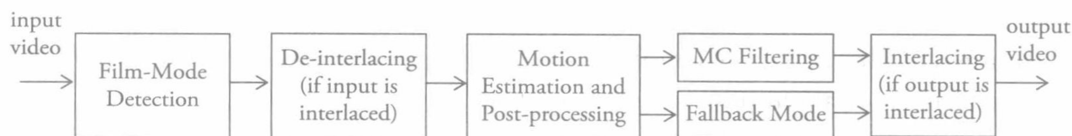



**Figure 6.19** Block diagram of an MC scan-rate conversion system.

#### *Film-Mode Detection*

If the input video has been converted to 50/60 Hz from an original 24 Hz movie by some pull-down method, this can be detected from successive frame similarity patterns, and the input video is converted back to 24 Hz progressive format prior to MC filtering.

## 6.3 Multi-Frame Noise Filtering

Fundamental concepts that make de-noising possible are self-similarity in the spatio-temporal domain and sparsity in a transform domain. Video contains significant self-similarity (redundancy) in the temporal dimension, which implies sparsity in transform domain (e.g., planar support in the Fourier domain for global translational motion as described in Section 6.1). As a result, inter-frame (multi-frame) noise filters can provide much better noise reduction than intra-frame filters. Motion-adaptive or MC filters can remove noise while avoiding spatial blurring of image detail [Dub 84, Sez 91, Boy 92, Liu 10, Mag 12].

### 6.3.1 Motion-Adaptive Noise Filtering

Motion-adaptive noise filters do not perform explicit motion estimation. They are applied over a fixed spatio-temporal support at each pixel. We start this section by discussing direct filtering where there is no adaptivity at all, or the adaptivity is implicit in the filter design. Next, we discuss filter structures, where some coefficients vary as a function of a so-called "motion-detection" signal.

#### Direct or Implicitly Motion-Adaptive Temporal Filtering

The simplest form of direct temporal filtering is frame averaging, where we average pixels occupying the same spatial coordinates in consecutive frames. Direct temporal averaging provides good results in the stationary parts of a frame, because averaging multiple observations of essentially the same pixel in different frames eliminates noise while preserving image detail. It is well known that direct averaging corresponds to maximum-likelihood estimation when there is no motion, assuming white, Gaussian noise, and reduces the variance of the noise by a factor of $N$, where $N$ is the number of samples [Uns 90]. It follows that in pure temporal filtering a large number of frames may be needed for noise reduction depending on the SNR. Spatio-temporal filtering provides a compromise between the number of frames needed for noise reduction and the amount of spatial blurring. Direct temporal averaging is not

motion-adaptive; hence, it causes smearing and chrominance separation in moving areas in the same way that direct spatial averaging leads to blurring around edges. A fundamental question that arises is how to distinguish temporal variations due to motion from those due to noise, which requires modeling motion.

Motion-adaptive filtering is the temporal counterpart of edge-preserving spatial filtering in that frame-to-frame motion gives rise to temporal edges. It follows that spatio-temporal noise filters that adapt to motion can be obtained by using structures similar to those of the edge-preserving filters. Implicitly motion-adaptive filters include directional filters and order statistic filters, such as median, weighted median, and multi-stage median filters [Arc 91]. For example, Martinez and Lim [Mar 85] proposed a cascade of five 1D FIR linear minimum mean-square error (LMMSE) estimators over a set of five hypothesized motion trajectories at each pixel that correspond to no motion, motion in the $+x_1$ direction, motion in the $-x_1$ direction, motion in the $+x_2$ direction, and motion in the $-x_2$ direction. Due to the adaptive nature of the LMMSE estimator, filtering is effective only along hypothesized trajectories that are close to actual ones.

### Motion-Detection Based Filtering

In motion-detection based filters, the selected filter structure has parameters that can be tuned according to a motion-detection signal, such as the frame difference. Both FIR and IIR filter structures can be employed in motion-adaptive filtering. A simple example of a motion-adaptive FIR filter is given by

$$\hat{s}[n_1, n_2, k] = (1-\gamma)g[n_1, n_2, k] + \gamma g[n_1, n_2, k-1] \tag{6.17}$$

and that of an IIR filter by

$$\hat{s}[n_1, n_2, k] = (1-\gamma)g[n_1, n_2, k] + \gamma \hat{s}[n_1, n_2, k-1] \tag{6.18}$$

where

$$\gamma = max\left\{0, \frac{1}{2} - \alpha\left|g[n_1, n_2, k] - g[n_1, n_2, k-1]\right|\right\} \tag{6.19}$$

is the motion-detection signal and $\alpha$ is a scaling constant. Observe that these filters tend to turn off filtering when a large motion is detected in an attempt to prevent artifacts. The FIR structure has limited noise-reduction ability, especially when used as a purely temporal filter with a small number of frames, because the reduction in

the noise variance is proportional to the number of samples in the filter support. IIR filters are more effective, but they generally cause Fourier-phase distortions.

Implementations of noise filters based on the above structures generally differ in the way they compute the motion-detection signal [Den 80].

### 6.3.2 Motion-Compensated Noise Filtering

The MC filtering approach is based on the assumption that the variation of the pixel gray levels over any motion trajectory $\mathbf{c}(\tau; x_1, x_2, t)$ is due mainly to noise. The motion trajectory $\mathbf{c}(\tau; x_1, x_2, t)$ is a continuous-valued vector function returning the coordinates of a point at time $\tau$ that corresponds to the point $(x_1, x_2)$ at time $t$. Thus, noise in both the stationary and moving areas of the image can effectively be reduced by low-pass filtering over the respective motion trajectory at each pixel. MC filters differ according to: i) the motion estimation method, ii) the support of the filter, (e.g., temporal vs. spatio-temporal), and iii) the filter structure (e.g., FIR vs. IIR, adaptive vs. non-adaptive).

The concept and estimation of a motion trajectory are illustrated in Figure 6.20. Suppose we wish to filter frame $k$ using $N$ frames centered about frame $k$, given by $k-M, \ldots, k-1, k, k+1, \ldots, k+M$, where $N=2M+1$. The first step is to estimate a discrete-motion trajectory $c(l; n_1, n_2, k)$, $l=k-M, \ldots, k-1, k, k+1, \ldots, k+M$, at each pixel $(n_1, n_2)$ of frame $k$. The function $c(l; n_1, n_2, k)$ is a continuous-valued vector function returning the $(x_1, x_2)$ coordinates of a point in frame $l$, which corresponds to the pixel $(n_1, n_2)$ of the $k$th frame. The discrete-motion trajectory is depicted by the solid line in Figure 6.20 for the case of $N = 5$ frames. In estimating

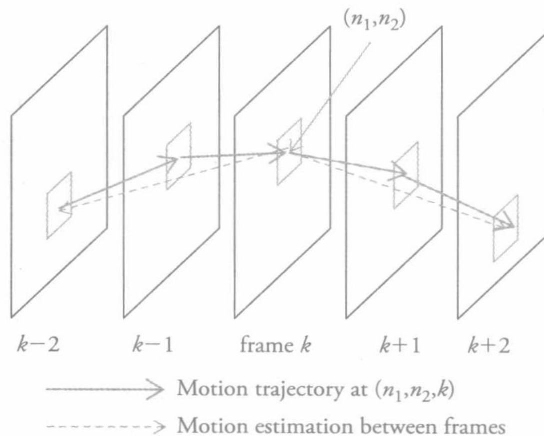



**Figure 6.20** Estimation of the motion trajectory ($M=2$, $N=5$).

the trajectory, the displacement vectors are usually estimated in reference to the frame $k$ as indicated by the dotted lines. The trajectory in general passes through sub-pixel locations, where the intensities can be determined via sub-pixel interpolation. The support $\boldsymbol{S}_{(n_1,n_2,k)}$ of an MC spatio-temporal filter is defined as the union of pre-determined spatial neighborhoods (e.g., 3×3 regions) centered about the pixel (sub-pixel) locations along the motion trajectory. In temporal filtering, the filter support $\boldsymbol{S}_{(n_1,n_2,k)}$ coincides with the motion trajectory $\boldsymbol{c}(l; n_1, n_2, k)$. Clearly, the effectiveness of MC spatio-temporal filtering is related to the accuracy of the motion estimates.

Various filtering techniques, ranging from averaging to more sophisticated adaptive filtering, can be employed given the MC filter support. In the ideal case, where the motion estimation is perfect, direct averaging of image intensities along a motion trajectory provides effective noise reduction [Mar 85]. In practice, motion estimation is hardly ever perfect due to noise, occlusions, and sudden scene changes, as well as changing camera views. As a result, image intensities over an estimated motion trajectory may not necessarily correspond to the same image structure, and temporal averaging may yield artifacts, hence the need for adaptive filter structures over the MC filter support.

We review two early adaptive MC filters: the MC-LMMSE filter, which turns filtering off whenever non-uniformity is detected within the MC support, and the MC adaptive weighted averaging (AWA) filter, which is a variation of bi-lateral filtering that weighs down the effect of outliers causing the non-uniformity. We also review two more recent MC filters that are extensions of the non-local means filter and BM3D filter studied in Chapter 3 for MC video de-noising.

### Spatio-Temporal Adaptive LMMSE Filtering

The MC adaptive LMMSE filter [Sam 85, Sez 91, Ozk 93] is an extension of the edge-preserving spatial filter that was presented in Section 3.5.3 for the spatio-temporal domain, where the local spatial statistics are replaced by their spatio-temporal counterparts. Then, the estimate of the pixel value at $(n_1, n_2, k)$ is given by

$$\hat{s}[n_1,n_2,k] = \mu_s[n_1,n_2,k] + \frac{\sigma_s^2(n_1,n_2,k)}{\sigma_s^2(n_1,n_2,k)+\sigma_v^2}\left(g[n_1,n_2,k] - \mu_g[n_1,n_2,k]\right) \quad (6.20)$$

where $\mu_.(n_1, n_2, k)$ and $\sigma_.^2(n_1, n_2, k)$ denote the ensemble mean and variance of the corresponding signal, respectively. Depending on whether we use spatio-temporal or temporal statistics, this filter will be referred to as the LMMSE-ST or the LMMSE-T filter, respectively.

We assume that signal-dependent noise can be modeled as

$$v(n_1, n_2, k) = s^{\alpha}(n_1, n_2, k) u(n_1, n_2, k)$$

where $u(n_1, n_2, k)$ is a wide-sense stationary process with zero-mean, which is independent of the signal, and $\alpha$ is a real number. The film grain noise is commonly modeled with $\alpha = 1/3$, and for signal-independent noise we have $\alpha = 0$. Under this model, $\mu_s(n_1, n_2, k) = \mu_g(n_1, n_2, k)$ and $\sigma_s^2(n_1, n_2, k) = \sigma_g^2(n_1, n_2, k) - \sigma_v^2$, where $\sigma_v^2$ denotes the variance of the noise process.

In practice, the ensemble mean $\mu_g(n_1, n_2, k)$ and variance $\sigma_g^2(n_1, n_2, k)$ are replaced with the sample mean $\hat{\mu}_g(n_1, n_2, k)$ and variance $\hat{\sigma}_g^2(n_1, n_2, k)$, which are computed within the support $\boldsymbol{S}_{(n_1, n_2, k)}$ as

$$\hat{\mu}_g(n_1, n_2, k) = \frac{1}{L} \sum\nolimits_{(i_1, i_2, l) \in S_{n_1, n_2, k}} g[i_1, i_2, l] \tag{6.21}$$

and

$$\hat{\sigma}_g^2[n_1, n_2, k] = \frac{1}{L} \sum\nolimits_{(i_1, i_2, l) \in S_{n_1, n_2, k}} \left( g[i_1, i_2, l] - \hat{\mu}_g[n_1, n_2, k] \right)^2 \tag{6.22}$$

where $L$ is the number of pixels in $\boldsymbol{S}_{(n_1, n_2, k)}$. Then

$$\hat{\sigma}_s^2(n_1, n_2, k) = \max \left\{ \hat{\sigma}_g^2(n_1, n_2, k) - \hat{\sigma}_v^2, 0 \right\} \tag{6.23}$$

in order to avoid the possibility of a negative variance estimate.

Substituting these estimates into the filter expression (6.20), we have

$$\hat{s}[n_1, n_2, k] = \frac{\hat{\sigma}_s^2(n_1, n_2, k)}{\hat{\sigma}_s^2(n_1, n_2, k) + \hat{\sigma}_v^2} g[n_1, n_2, k] + \frac{\hat{\sigma}_v^2}{\hat{\sigma}_s^2(n_1, n_2, k) + \hat{\sigma}_v^2} \hat{\mu}_g[n_1, n_2, k] \tag{6.24}$$

The adaptive nature of the filter can be observed from (6.24). When the spatio-temporal signal variance is much smaller than the noise variance, $\hat{\sigma}_s^2(n_1, n_2, k) \approx 0$, i.e., the support $\boldsymbol{S}_{(n_1, n_2, k)}$ is uniform, the estimate approaches the spatio-temporal mean, $\hat{\mu}_g(n_1, n_2, k)$. At the other extreme, when the spatio-temporal signal variance is much larger than the noise variance, $\hat{\sigma}_s^2(n_1, n_2, k) \gg \hat{\sigma}_v^2$, due to poor motion estimation or the presence of sharp spatial edges in $\boldsymbol{S}_{(n_1, n_2, k)}$, the estimate approaches the noisy image value to avoid blurring.

A drawback of the adaptive LMMSE filter is that it turns the filtering down even if there are a few outlier pixels in the filter support, thus leaving noise in the filtered

image. An alternative implementation, called "the switched LMMSE filter," may maintain the advantages of both the spatio-temporal and the temporal LMMSE filtering by switching between a selected set of temporal and spatio-temporal supports at each pixel, depending on which support is the most uniform [Ozk 93]. If the variance $\hat{\sigma}_g^2(n_1,n_2,k)$ computed over $\boldsymbol{S}_{(n_1,n_2,k)}$ is less than the noise variance, then the filtering is performed using this support; otherwise, the largest support over which $\hat{\sigma}_g^2(n_1,n_2,k)$ is less than the noise variance is selected. In the next section, we introduce an adaptive weighted averaging (AWA) filter that employs an implicit mechanism for selecting the most uniform subset within $\boldsymbol{S}_{(n_1,n_2,k)}$ for filtering.

### Adaptive-Weighted-Averaging Filter

The adaptive-weighted-averaging (AWA) filter is a variation of the bi-lateral or sigma filter, which computes a weighted average of the intensity values within the spatio-temporal support along the motion trajectory. The weights are determined by optimizing a criterion functional, and they vary with the accuracy of motion estimation as well as the spatial uniformity of the region around the motion trajectory. In the case of sufficiently accurate motion estimation across the entire trajectory and spatial uniformity, image values within the spatio-temporal filter support attain equal weights, and the AWA filter performs direct spatio-temporal averaging. When the value of a pixel within the spatio-temporal filter support deviates from the value of the pixel to be filtered by more than a threshold, its weight decreases, shifting the emphasis to other pixels within the support that better match the pixel of interest. The AWA filter is therefore particularly well suited for efficient filtering of sequences containing segments with varying scene contents due, for example, to rapid zooming and changes in the view of the camera.

The AWA filter can be defined by

$$\hat{s}[n_1,n_2,k]=\sum_{(i_1,i_2,l)\in S_{n_1,n_2,k}} w(i_1,i_2,l)\,g[i_1,i_2,l] \tag{6.25}$$

where

$$w(i_1,i_2,l)=\frac{K(n_1,n_2,k)}{1+a\max\left\{\varepsilon^2,(g[n_1,n_2,k]-g[i_1,i_2,l])^2\right\}}$$

are the weights within the support $\boldsymbol{S}_{(n_1,n_2,k)}$ along the motion trajectory and $K(n_1,n_2,k)$ is a normalization constant, given by

$$K(n_1,n_2,k)=\left[\sum_{(i_1,i_2,l)\in S_{n_1,n_2,k}} \frac{1}{1+a\max\left\{\varepsilon^2,(g[n_1,n_2,k]-g[i_1,i_2,l])^2\right\}}\right]^{-1}$$

The quantities $a>0$ and $\varepsilon$ are the parameters of the filter. These parameters are determined according to the following principles:

1. When the differences in the intensities of pixels within the spatio-temporal support are merely due to noise, it is desirable that the weighted averaging reduces to direct averaging. This can be achieved by selecting the parameter $\varepsilon^2$ appropriately. Note that if the square of the differences is less than $\varepsilon^2$, then all the weights attain the same value $K/(1+a\,\varepsilon^2)=1/L$ and $\hat{s}[n_1, n_2, k]$ reduces to direct averaging. We set the value of $\varepsilon^2$ equal to two times the value of the noise variance, i.e., the expected value of the square of the difference between two pixel values that differ due only to the presence of noise.
2. If the square of the difference between the values $g[n_1, n_2, k]$ and $g[i_1, i_2, l]$ for a particular $(i_1, i_2, l) \in \boldsymbol{S}_{(n_1, n_2, k)}$ is larger than $\varepsilon^2$, then the contribution of $g[i_1, i_2, l]$ is weighted down by $w(i_1, i_2; l) < w(n_1, n_2, k) = K/(1+a\,\varepsilon^2)$. The "penalty" parameter $a$ determines the sensitivity of the weight to the squared difference $(g[n_1, n_2, k]-g[i_1, i_2, l])^2$. It is usually set equal to unity.

The effect of the penalty parameter $a$ on the performance of the AWA filter can be best visualized considering a special case where one of the frames within the filter support is substantially different from the rest. In the extreme, when $a=0$, all weights are equal. That is, there is no penalty for a mismatch, and the AWA filter performs direct averaging. However, for $a$ large, the weights for the $2M$ "matching" frames are equal, whereas the weight of the non-matching frame approaches zero. Generally speaking, the AWA filter takes the form of a "limited-amplitude averager," where those pixels whose intensities differ from that of the center pixel by no more than $\pm\varepsilon$ are averaged. A similar algorithm is $K$-nearest neighbor averaging [Dav 78], where the average of $K$ pixels within a certain window whose values are closest to the value of the pixel of interest are computed.

The noise variance appears directly in the LMMSE filter expression, whereas in the AWA filter, it is used to define the filter parameter $\varepsilon^2$, which is typically set equal to twice the estimated noise variance. Inspection of the results suggests that the spatio-temporal filters provide better noise reduction than the respective temporal filters, at the expense of introducing some blur. The switched LMMSE filter strikes the best balance between noise reduction and retaining image sharpness. The visual difference between the results of the LMMSE and AWA filters is insignificant if there are no noise outliers or sudden scene changes involved. It has been shown that the AWA filter outperforms the LMMSE filter, especially in cases of low-input SNR and abruptly varying scene content [Ozk 93].

### Temporally Coherent NLM Filtering

Non-local means (NLM) filtering for image de-noising (see Chapter 3) has been extended to temporally coherent filtering by exploiting self-similarity in the spatial and temporal dimensions [Liu 10]. The authors introduce approximate $K$-nearest neighbor patch matching to find a set of supporting patches from the current frame and temporally adjacent frames that are similar to the current patch. The proposed search method has lower search complexity to allow for searching for similar patches over the entire frame. They also estimate the noise level at each frame for noise-adaptive de-noising. The authors argue that robust motion estimation and filtering over temporally coherent patches along the motion path are essential for high-quality video de-noising.

### BM4D Filtering

Block matching 4D (BM4D) extends the powerful collaborative filtering paradigm of BM3D for image de-noising (see Chapter 3) to video filtering by exploiting both non-local spatial and temporal self-similarity and sparsity in transform domain [Mag 12]. It is well known that the similarity of blocks along the motion trajectory is stronger than the non-local similarity existing within an individual frame even in the presence of fast motion. An earlier extension of BM3D to video de-noising, called V-BM3D, groups similar 2D blocks extracted from a set of consecutive frames into 3D arrays regardless of whether they come from temporal similarity or the non-local spatial similarity. In contrast, V-BM4D groups mutually similar spatio-temporal volumes, a collection of 3D structures formed by a sequence of blocks of video following a specific trajectory, computed according to a non-local search procedure. Hence, groups in V-BM4D are 4D stacks of 3D volumes, and the collaborative filtering is then performed via a separable 4D spatio-temporal transform. V-BM4D provides state-of-the-art video de-noising.

## 6.4 Multi-Frame Restoration

We discussed single frame (still) image restoration in Section 3.6. This section extends the formulation of the image-restoration problem to the case when we have a correlated sequence of blurred images, which are degraded by possibly different point-spread functions (PSF). The temporal dimension can be used to both estimate the PSF and to improve the quality of restored images. For example, in linear-motion blur, the extent of the spatial spread of a point source at a given pixel during the aperture time can be computed from an estimate of the frame-to-frame velocity (motion vector) at that pixel, provided that the shutter speed of the camera is known

[Tru 92]. Furthermore, if the blur PSF changes from frame-to-frame, the locations of zero-crossing of the blur frequency response vary from frame-to-frame enabling extraction of more information from a collection of frames than can be extracted from any single frame.

## 6.4.1 Multi-Frame Modeling

Suppose we have $L$ frames of video, each blurred by possibly a different spatially invariant PSF, $h_k[n_1, n_2]$, $k=1, \ldots, L$. The vector-matrix model can be extended to multi-frame modeling over $L$ frames as

$$\mathbf{g} = \boldsymbol{D}\mathbf{s} + \mathbf{v} \tag{6.26}$$

where

$$\mathbf{g} = \begin{bmatrix} \mathbf{g}_1 \\ \vdots \\ \mathbf{g}_L \end{bmatrix}, \ \mathbf{s} = \begin{bmatrix} \mathbf{s}_1 \\ \vdots \\ \mathbf{s}_L \end{bmatrix}, \ \mathbf{v} = \begin{bmatrix} \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_L \end{bmatrix}$$

are $N^2L \times 1$ vectors representing the observed, ideal, and noise frames, respectively, stacked as multi-frame vectors, and

$$\boldsymbol{D} = \begin{bmatrix} \mathbf{D}_1 & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{D}_L \end{bmatrix}$$

is an $N^2L \times N^2L$ matrix representing the multi-frame blur operator. Observe that the multi-frame blur matrix $\boldsymbol{D}$ is block-diagonal, indicating no temporal blurring.

## 6.4.2 Multi-Frame Wiener Restoration

We employ a multi-frame Wiener deconvolution framework to exploit the temporal correlation between frames for spatially shift-invariant but temporally shift-varying blurs. That is, we have a shift-invariant 2D blur at each frame, but the blur PSF can change from frame-to-frame. Extension to the case of multi-frame spatially shift-varying restoration is discussed in Section 6.5.4 within the POCS framework.

Applying the CLS filter given by (3.107) to the observation equation (6.26) with $\mathbf{L}^T\mathbf{L} = \boldsymbol{R}_s^{-1}\boldsymbol{R}_v$, we obtain the Wiener estimate $\hat{\mathbf{s}}$ of the $L$ frames $\mathbf{s}$ as

$$\hat{\mathbf{s}} = \left(\boldsymbol{D}^{\mathrm{T}}\boldsymbol{D} + \boldsymbol{R}_s^{-1}\boldsymbol{R}_v\right)^{-1}\boldsymbol{D}^T\mathbf{g} \tag{6.27}$$

where

$$\hat{\mathbf{s}} = \begin{bmatrix} \hat{\mathbf{s}}_1 \\ \vdots \\ \hat{\mathbf{s}}_L \end{bmatrix} \quad \boldsymbol{R}_s = \begin{bmatrix} \mathbf{R}_{s;11} & \cdots & \mathbf{R}_{s;1L} \\ \vdots & \ddots & \vdots \\ \mathbf{R}_{s;L1} & \cdots & \mathbf{R}_{s;LL} \end{bmatrix} \text{ and } \boldsymbol{R}_v = \begin{bmatrix} \mathbf{R}_{v;11} & \cdots & \mathbf{R}_{v;1L} \\ \vdots & \ddots & \vdots \\ \mathbf{R}_{v;L1} & \cdots & \mathbf{R}_{v;LL} \end{bmatrix}$$

in which $\mathbf{R}_{s;ij} = \boldsymbol{E}\left\{\mathbf{s}_i \mathbf{s}_j^T\right\}$ and $\mathbf{R}_{v;ij} = \boldsymbol{E}\left\{\mathbf{v}_i \mathbf{v}_j^T\right\}$, $i,j = 1, 2, \ldots, L$. Note that if $\mathbf{R}_{s;ij} = \mathbf{0}$ for $i \neq j$, $i,j = 1, 2, \ldots, L$, then the multi-frame estimate becomes equivalent to stacking the $L$ single-frame estimates obtained individually.

Again, direct computation of $\hat{\mathbf{s}}$ requires the inversion of the $N^2L \times N^2L$ matrix in (6.27). Because the blur PSF is not necessarily the same in each frame, and the image correlations are generally not shift-invariant in the temporal direction, the matrices $\boldsymbol{D}$, $\boldsymbol{R}_s$, and $\boldsymbol{R}_v$ are not block-Toeplitz; thus, a 3D-DFT would not diagonalize them. However, each $\mathbf{D}_k$ is block-Toeplitz. Furthermore, assuming each image and noise frame is wide-sense stationary in the 2D plane, $\mathbf{R}_{s;ij}$ and $\mathbf{R}_{v;ij}$ are also block-Toeplitz. Approximating the block-Toeplitz sub-matrices $\mathbf{D}_i$, $\mathbf{R}_{s;ij}$, and $\mathbf{R}_{v;ij}$ by block-circulant sub-matrices, each sub-matrix can be diagonalized by separate 2D-DFT operations in an attempt to simplify matrix calculations. To this effect, we define the $N^2L \times N^2L$ transformation matrix

$$\boldsymbol{W}^{-1} = \begin{bmatrix} \mathbf{W}^{-1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \mathbf{W}^{-1} \end{bmatrix}$$

where the $N^2 \times N^2$ matrix $\boldsymbol{W}^{-1}$ is the 2D-DFT operator defined previously. Note that the matrix operator $\boldsymbol{W}^{-1}$, when applied to $\boldsymbol{s}$, stacks the vectors formed by 2D-DFTs of the individual frames, but it is not the 3D-DFT operator.

We pre-multiply both sides of (6.27) with $\boldsymbol{W}^{-1}$ to obtain

$$\boldsymbol{W}^{-1}\hat{\boldsymbol{s}} = \left[\boldsymbol{W}^{-1}\boldsymbol{D}\boldsymbol{W}\boldsymbol{W}^{-1}\boldsymbol{D}^{\mathrm{T}}\boldsymbol{W} + (\boldsymbol{W}\boldsymbol{R}_s\boldsymbol{W}^{-1})^{-1}\boldsymbol{W}^{-1}\boldsymbol{R}_v\boldsymbol{W}\right]^{-1}\boldsymbol{W}^{-1}\boldsymbol{D}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{W}^{-1}\mathbf{g}$$

which can be expressed as

$$\hat{\mathbf{S}} = \boldsymbol{Q}^{-1}\boldsymbol{H}^{*}\boldsymbol{G} \tag{6.28}$$

where $\boldsymbol{G} = \boldsymbol{W}^{-1}\,\mathbf{g}$ is an $N^2L$ vector and $\boldsymbol{P}_{\mathrm{s}} = \boldsymbol{W}^{-1}\,\boldsymbol{R}_{\mathrm{s}}\boldsymbol{W}$, $\boldsymbol{H}^{*} = \boldsymbol{W}^{-1}\,\boldsymbol{D}^{\mathrm{T}}\boldsymbol{W}$, and $\boldsymbol{Q} = \boldsymbol{H}\boldsymbol{H}^{*} + \boldsymbol{P}_{\mathrm{s}}$ are $N^2L \times N^2L$ block matrices with $N^2 \times N^2$ blocks, where each block is a diagonal matrix.

We present two approaches for the computation of $\boldsymbol{Q}^{-1}$: a general algorithm called the cross-correlated multi-frame (CCMF) Wiener filter, which requires the auto- and cross-power spectra of the frames, and a specific algorithm called the motion-compensated multi-frame filter (MCMF), which applies to the special case of global, constant-velocity motion when a closed-form solution becomes possible.

#### Cross-Correlated Multi-Frame Filter

It was shown in [Ozk 92] that $\boldsymbol{Q}^{-1}$ is also a block matrix with diagonal blocks, and the elements of $\boldsymbol{Q}^{-1}$ can be computed by inverting $N^2$ matrices each $L \times L$. This derivation will not be given here. The inversion of $L \times L$ matrices can be performed in parallel. Furthermore, if $L$ is sufficiently small, the $L \times L$ matrices can be inverted using an analytic inversion formula.

The computation of the multi-frame Wiener estimate requires knowledge of the covariance matrices $\mathbf{R}_s$ and $\mathbf{R}_v$. We assume the noise is spatio-temporally white; thus, the matrix $\mathbf{R}_v$ is diagonal with all diagonal entries equal to $\sigma_v^2$, although the formulation allows for any noise covariance matrix. The estimation of the multi-frame ideal video-covariance matrix $\boldsymbol{R}_s$ can be performed by either the periodogram method or 3D-AR modeling [Ozk 92]. Once $\boldsymbol{Q}^{-1}$ is determined using the discussed scheme, the Wiener estimate can be computed from (6.28).

#### Motion-Compensated Multi-Frame Filter

Assuming that the auto-power spectra of all frames are the same and there is global (circular) motion, the cross spectra of the frames are related by a phase factor determined by the motion. Given the motion vectors (one for each frame) and the auto-power spectrum of the reference frame, the matrix $\boldsymbol{Q}^{-1}$ (hence, the Wiener estimate) can be computed analytically using the Sherman–Morrison formula without an explicit matrix inversion. The interested reader is referred to [Ozk 92] for further details.

Other approaches for multi-frame image restoration include iterative non-linear optimization formulations with application to imaging through atmospheric turbulence (Section 3.8 of [Bov 00]) and reduced-order model Kalman filtering for progressive and interlaced video [Pat 98].

## 6.5 Multi-Frame Super-Resolution

Super-resolution (SR) methods can be broadly classified as recognition-based or example-based single-frame methods vs. reconstruction-based multi-frame methods. The former was briefly discussed in Chapter 3 in the context of nonlinear

image interpolation. This section presents multi-frame reconstruction-based SR methods. We start by discussing how the reconstruction-based SR problem differs from interpolation and restoration problems and what makes SR possible in Section 6.5.1. Section 6.5.2 presents a model that relates the observed low-resolution (LR) frames with aliasing to a hypothetical higher-resolution reference image. The SR-reconstruction problem addresses recovery of this higher-resolution image from multiple LR frames with aliasing and sub-pixel motion. An early frequency-domain solution for a simple special case is described in Section 6.5.3. More recent spatio-temporal domain solutions to the general SR problem are presented in Section 6.5.4. The reader is referred to [Par 03] for a general overview of SR-reconstruction methods.

## 6.5.1 What Is Super-Resolution?

Super-resolution (SR) is the process of reconstruction of spatial frequencies beyond half of the Nyquist sampling rate that are clearly not available in a sampled image. Assuming the frequency is normalized by the sampling rate, such that the highest horizontal and vertical spatial frequencies in the input image are $\pi$, the input image can be first up-sampled by a factor of $L$, shrinking the highest spatial frequencies in the up-sampled image to $\pi/L$ (see Section 3.2). Now, the SR problem can be defined as estimating those frequencies in the region $(\pi/L<\omega_1<\pi) \cup (\pi/L<\omega_2<\pi)$ for the up-sampled image. Assuming that the input image is $N_1 \times N_2$, the output image is $LN_1 \times LN_2$, and there are $L^2-1$ unknown pixels for each input pixel. Hence, the problem is highly ill-posed (under-determined), and there exist infinitely many possible solutions in the absence of a strong image-formation model and/or *a priori* information about the high-resolution (HR) image that constrains the solution space.

SR methods can be classified as recognition/example-based vs. reconstruction-based methods. The former methods aim to learn or recognize the missing high-frequency patterns from a set of examples or a dictionary [Bak 02, Fre 02], which amounts to regularization of the problem by "learned" *a priori* models for the desired HR image. Although these methods yield images that are sharper than can be obtained by linear interpolation, they do not attempt to model either the aliasing or blurring present in the observed LR images. The reconstruction-based methods exploit the aliasing present in LR images, since no imaging sensor employs perfect anti-aliasing, by modeling LR image formation accounting for aliasing and sub-pixel motion between LR and HR grids (see "What makes super-resolution possible?" below).

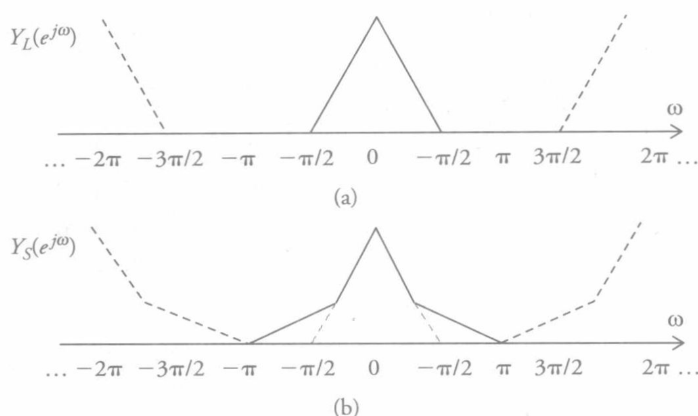



**Figure 6.21** Fourier spectrum of an up-sampled digital signal by a factor of 2: (a) linear interpolation vs. (b) super-resolution.

### Super-Resolution vs. Image Interpolation

It is well-known that no new high-frequency information (i.e., not present in the input image) can be generated by linear shift-invariant interpolation including the ideal band-limited interpolation. Given a full-bandwidth (critically sampled) input image occupying the entire frequency range $-\pi<|\omega|<\pi$, we know from Section 3.2.2 that linear interpolation by a factor of $L$ contracts the output spectrum to $-\frac{\pi}{L}<|\omega|<\frac{\pi}{L}$. This is illustrated in Figure 6.21(a), where the spectrum of a linearly interpolated signal ($L=2$) satisfies $Y_L(e^{j\omega})=0$, for $\frac{\pi}{2}<|\omega|<\pi$. Note that the frequency variable $\omega$ of the up-sampled signal is normalized with the higher sampling rate, such that the frequency $\pi/L$ for the output corresponds to the frequency $\pi$ for the input. SR methods aim to recover (reconstruct) the missing high-frequency band as illustrated in Figure 6.21(b); hence, they reconstruct not only a larger image in size but also with higher resolution or definition (with higher frequency content).

### Super-Resolution vs. Image Restoration

Image restoration does not involve up-sampling an image; hence, no new frequency band is created. Image-restoration filters correct the Fourier magnitude/phase of optically distorted images only within the original input image-frequency range given the optical-transfer function of the distortion (blur).

### What Makes Super-Resolution Possible?

SR from multiple LR frames is a better-posed problem than single-frame SR since each LR frame with sub-pixel motion potentially contains novel information about

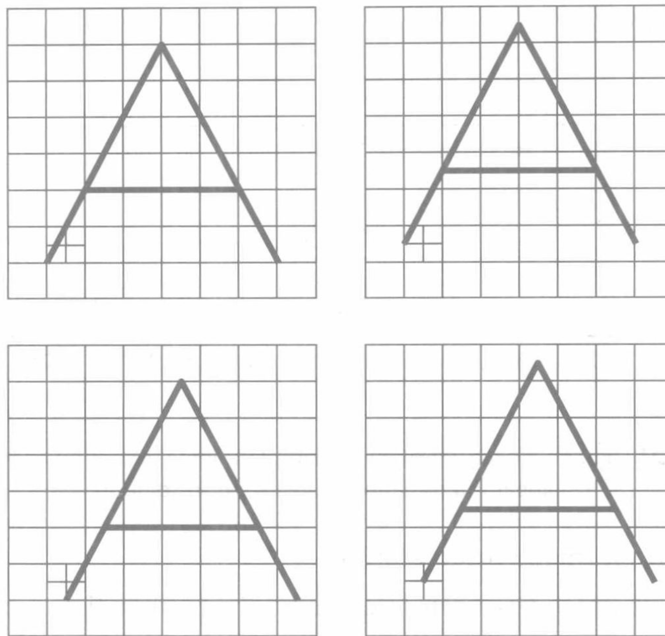


**Figure 6.22** Sub-pixel displacement between frames allows higher-resolution reconstruction.

the desired HR image, provided that we can estimate sub-pixel motion between multiple LR frames accurately. In addition, the presence of aliasing in LR images is essential for recovery of missing high-frequency information (see Section 6.2.1). Indeed, aliasing is a representation of the missing high-frequency information folded over existing frequencies; i.e., the high-frequency content of the desired HR image is embedded in the available (low) frequency band of each LR image. Hence, it is a combination of i) the presence of aliasing in LR images, and ii) sub-pixel displacements between multiple images (frames) that make super-resolution possible.

Four sub-pixel-shifted LR frames are illustrated in Figure 6.22, where large squares denote low-resolution pixels, whose intensity is proportional to the average brightness within each square. Each image is shifted by a half-pixel to the right and/or half-pixel up in the LR image coordinates and, hence, contains new observations. Note that if they were shifted by an integer pixel in both directions, then all four images would consist of exactly the same pixels (just displaced); hence, multiple frames would contain no new information. When $L=2$, we estimate four smaller (HR) pixels for each LR pixel. The estimated HR image should be consistent with all four LR images according to the image capture model.

**Limits of Super-Resolution**

How fast the SR reconstruction deteriorates as the magnification factor increases and whether there is a fundamental limit on the magnification factor that can be achieved have been analyzed based on perturbation theory of linear system of equations and the condition number of the coefficient matrix [Bak 02, Lin 04]. However, it is worth noting that these studies do not consider i) the amount of aliasing in the LR images, ii) the accuracy of registration, and iii) the SNR of the input LR images explicitly in their analysis.

## 6.5.2 Modeling Low-Resolution Sampling

Most consumer video cameras can record frames at a resolution lower than desirable. This is related to some physical limitations, such as finite-sensor-cell area and finite aperture time. Although high-resolution professional cameras exist, these may be too expensive and/or unsuitable for mobile-imaging applications. In the following, we first present a model that relates observed low-resolution (LR) sampled video frames with aliasing to the underlying continuous video. Next, the LR frames are expressed in terms of a hypothetical high-resolution (HR) reference video frame. The super-resolution problem is posed as a reconstruction of this HR frame from a number of observed LR frames that are sub-pixel registered.

**Continuous-Discrete Model**

A comprehensive model of LR video acquisition should include the effects of finite-sensor-cell area and finite-aperture time. LR images suffer from a combination of blurring due to spatial integration at the sensor surface (due to finite cell area), modeled by a shift-invariant spatial PSF $h_a(x_1, x_2)$, and aliasing due to sub-Nyquist sampling. In addition, relative motion between the scene and the camera during the aperture time gives rise (due to temporal integration) to shift-varying spatial blurring. The relative motion may be due to camera motion, as in camera pan or a camera mounted on a moving vehicle, or the motion of objects in a scene, which is especially significant in high-action movies and sports videos. Recall that it is often this motion that gives rise to temporal variation of the spatial-intensity distribution. Images may also suffer from out-of-focus blur, which is modeled by a shift-invariant spatial PSF $h_o(x_1, x_2)$ (ignoring the depth-of-field) and additive noise.

A block diagram of the continuous-input, discrete-output model of a low-resolution video-acquisition process is depicted in Figure 6.23. The first sub-system

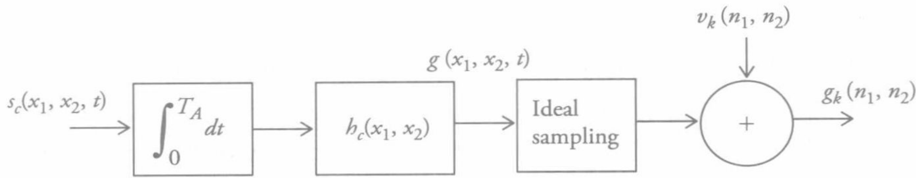



**Figure 6.23** Block diagram of video acquisition.

models temporal integration during the aperture time $T_A$, resulting in a blurred video (due to relative motion) given by

$$g_m(x_1, x_2, t) = \frac{1}{T_A} \int_0^{T_A} s_c(x_1, x_2, t - \gamma)\, d\gamma \tag{6.29a}$$

where $s_c(x_1, x_2, t)$ denotes the ideal continuous video that would be formed by an instantaneous aperture. The model (6.29a) can be interpreted by considering a stationary camera and mapping the effect of any camera motion to a relative scene motion. The next sub-system models the combined effects of integration at the sensor surface and any shift-invariant out-of-focus blur, resulting in a further blurred video given by

$$g(x_1, x_2, t) = h_c(x_1, x_2) ** g_m(x_1, x_2, t) \tag{6.29b}$$

where $h_c(x_1, x_2) = h_a(x_1, x_2) ** h_o(x_1, x_2)$ is the combined shift-invariant spatial PSF and $**$ denotes 2D convolution. The integration in the model over time (6.29a) can equivalently be written as a spatial integration at a single time instant $\tau$ using our ideal video source model

$$s_c(x_1, x_2, t) = s_c\big(c_1(\tau; x_1, x_2, t),\ c_2(\tau; x_1, x_2, t), \tau\big) \tag{6.30}$$

where $\mathbf{c}(\tau; x_1, x_2, t) = \big(c_1(\tau; x_1, x_2, t),\ c_2(\tau; x_1, x_2, t)\big)$ is the motion trajectory function that is defined in Section 6.1.1. Then, assuming the reference time $\tau$ is within the temporal span of all motion trajectories passing through $(x_1, x_2, t - \gamma)$, $0 < \gamma < T_A$, all $(x_1, x_2)$ within the time interval $(t - T_A, t)$ can be traced onto the plane at time $\tau$ using the source model (6.30) to perform spatial integration over these points. The corresponding spatial PSF is given by

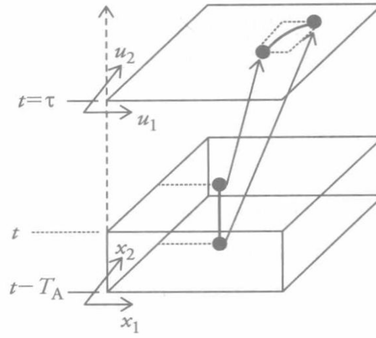



**Figure 6.24** Representation of the effect of motion during the aperture time (temporal integration) by a spatial PSF (spatial integration).

$$h_m(u_1,u_2,\tau;x_1,x_2,t) = \tag{6.31}$$
$$\begin{cases} \frac{1}{T_A} J(u_1,u_2,t)\delta\left(u_2 - c_2\left(\tau;x_1,x_2,c_1^{-1}(\tau;x_1,x_2,u_1)\right)\right) & \text{if } c_1(\tau;x_1,x_2,t-T_A) < u_1 < c_1(\tau;x_1,x_2,t) \\ 0 & \text{otherwise} \end{cases}$$

where $J(u_1, u_2, t)$ is the Jacobian of the change of variables and $c_1^{-1}(\tau;x_1,x_2,u_1)$ returns the time $t-T_A<t_0<t$ when $u_1=c_1(\tau;x_1,x_2,t_0)$. (See [Pat 97b] for a complete derivation of this PSF.) The spatial support of $h_m(u_1,u_2,\tau;x_1,x_2,t)$, which is a shift-variant PSF, is a path at the reference time $\tau$, which is depicted in Figure 6.24.

The path, mathematically expressed in Eqn. (6.31) by the 1D delta function within the interval $c_1(\tau;x_1,x_2,t-T_A)<u_1<c_1(\tau;x_1,x_2,t)$, is obtained by mapping all points $(x_1,x_2,t-\gamma)$, $0<\gamma<T_A$ (shown by the solid vertical line) onto time $\tau$. Therefore, combining the video acquisition model in Figure 6.23 with the source model (6.30), we can establish the relationship

$$g(x_1,x_2,t) = \iint s_c(u_1,u_2,\tau)h(u_1,u_2,\tau;x_1,x_2,t)\,du_1\,du_2 \tag{6.32}$$

between the observed intensity $g(x_1,x_2,t)$ and the ideal intensities $s_c(u_1,u_2,\tau)$ along the path $(u_1,u_2)=\mathbf{c}(\tau;x_1,x_2,t)$ at time $\tau$, where

$$h(u_1,u_2,\tau;x_1,x_2,t) = h_c(x_1,x_2) ** h_m(u_1,u_2,\tau;x_1,x_2,t)$$

denotes the effective shift-variant spatial PSF of the composite model.

The degraded video is then ideally sampled on a 3D low-resolution lattice $\Lambda_l$, and noise is added to form the observed discrete video

$$g_k(n_1,n_2) = g(x_1,x_2,t)\,\big|_{[x_1,x_2,t]^{\mathrm{T}} = \mathbf{V}_l[n_1,n_2,k]^{\mathrm{T}}} + v_k(n_1,n_2) \tag{6.33}$$

where $\mathbf{V}_l$ is the sampling matrix of the low-resolution lattice $\Lambda_l$. If we assume that $g(x_1, x_2, t)$ is bandlimited, but the sampling intervals in the $x_1$, $x_2$, and $t$ directions are larger than the corresponding Nyquist intervals, aliasing may result. It is essential that no anti-alias filtering be used prior to sampling in order to achieve super-resolution, which follows from the discussion in Section 6.2.1.

### Discrete-Discrete Model

Next, we relate a set of low-resolution observations $g_k(n_1, n_2)$ to the desired reference high-resolution frame to be reconstructed, which is defined as

$$\mathrm{s}(m_1,m_2) = s_c(x_1,x_2,t)\,\big|_{[x_1,x_2,t]^{\mathrm{T}} = \mathbf{V}_h[m_1,m_2,i]^{\mathrm{T}}} \tag{6.34}$$

where $\mathbf{V}_h$ is the sampling matrix of the high-resolution sampling grid. Let's assume that the high-resolution frame $\mathrm{s}(m_1, m_2)$ is sampled above the Nyquist rate, so that the continuous intensity pattern is more or less constant within each high-resolution pixel (cells depicted in Figure 6.25). Then, given the motion trajectory $\mathbf{c}(\tau; x_1, x_2, t)$ passing through $(n_1, n_2, k)$, we have, from (6.33) and (6.32),

$$g_k(n_1,n_2) \approx \mathrm{s}(m_1,m_2) \iint h(u_1,u_2,\tau;x_1,x_2,t)\,du_1\,du_2$$

where $[x_1, x_2, t]^{\mathrm{T}} = \mathbf{V}_1[n_1, n_2, k]^{\mathrm{T}}$, $[u_1, u_2, \tau]^{\mathrm{T}} = \mathbf{V}_h[m_1, m_2, i]^{\mathrm{T}}$, and $(u_1, u_2) = \mathrm{c}(\tau; x_1, x_2, t)$; i.e., $\tau$ is the time coordinate of the reference frame and $(m_1, m_2)$ denotes the spatial coordinates of the pixel in the high-resolution reference frame that matches $(n_1, n_2, k)$. Next, we define

$$h_k(m_1,m_2;n_1,n_2) = \iint h(u_1,u_2,\tau;x_1,x_2,t)\,du_1\,du_2$$

to arrive at our discrete-input (high-resolution video), discrete-output (observed low-resolution video) model, given by

$$g_k(n_1,n_2) = \textstyle\sum_{m_1}\sum_{m_2} \mathrm{s}(m_1,m_2)h_k(m_1,m_2;n_1,n_2) + v_k(n_1,n_2) \tag{6.35}$$

where the support of the summation over the high-resolution grid $(m_1, m_2)$ at a particular low-resolution sample $(n_1, n_2, k)$ is depicted in Figure 6.25.

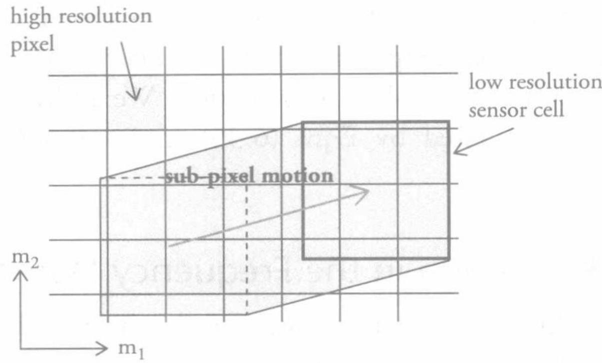



**Figure 6.25** Illustration of the discrete system PSF.

The size of the support in Figure 6.25 depends on the relative velocity of the scene with respect to the camera, the size of the support of the low-resolution sensor PSF $h_a(x_1, x_2)$ (depicted by the solid line, assuming no out-of-focus blur) with respect to the high-resolution grid, and whether there is any out-of-focus blur. Because the relative positions of low- and high-resolution pixels in general vary from pixel-to-pixel, the discrete sensor PSF is space-varying.

The model (6.35) establishes a relationship between the desired high-resolution frame and the observed low-resolution pixels from all frames $k$ that are connected to the desired frame by means of a motion trajectory. That is, each low-resolution observed pixel $(n_1, n_2, k)$ can be expressed as a linear combination of several high-resolution pixels from the desired frame provided that $(n_1, n_2, k)$ is connected to the desired frame by a motion trajectory. We assume that occlusion regions can be detected and pixels for which the model is invalid are excluded.

### Problem Interrelations

The super-resolution problem stated by (6.35) is a superset of other filtering problems that are discussed in this chapter as follows:

1. Noise filtering: Input and output lattices are identical $\Lambda_l \neq \Lambda_h$; sensor PSF is not taken into account, $h_a(x_1, x_2) = \delta(x_1, x_2)$; the camera aperture time is negligible, $T_A = 0$; and there is no optical blur, $h_o(x_1, x_2) = \delta(x_1, x_2)$.
2. Multi-frame/image restoration: Input and output lattices are identical, $\Lambda_l \neq \Lambda_h$; sensor PSF is not taken into account, i.e., we assume $h_a(x_1, x_2) = \delta(x_1, x_2)$.
3. Standards conversion: Input and output lattices are different, $\Lambda_l \neq \Lambda_h$, but sensor PSF is not taken into account, $h_a(x_1, x_2) = \delta(x_1, x_2)$; the camera aperture time is negligible, $T_A = 0$; there is no optical blur, $h_o(x_1, x_2) = \delta(x_1, x_2)$; and there is no noise.

We first present a frequency-domain solution to a special case (global translation) in Section 6.5.3, which helps the reader to understand why aliasing should be present in the LR images for SR reconstruction. We address the general super-resolution problem formulated by Eqn. (6.35) in Section 6.5.4 using spatial-domain methods.

### 6.5.3 Super-Resolution in the Frequency Domain

The frequency-domain method, first proposed by Tsai and Huang [Tsa 84], exploits the relationship between the continuous and discrete-Fourier transforms of the under-sampled frames in the special case of global sub-pixel shifts. If we let $s_0(x_1, x_2) \equiv s_c(x_1, x_2, 0)$ denote the reference frame, and assume zero aperture time ($T_A=0$) and rectangular sampling with the sampling interval $\Delta$ in both directions, then the continuous-input, discrete-output model given by Eqn. (6.33) simplifies as

$$g_k(n_1, n_2) = \sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1} [s_0(x_1 - \alpha_k, x_2 - \beta_k) ** h_c(x_1, x_2)]\ \delta(x_1 - n_1\Delta, x_2 - n_2\Delta) + v_k(n_1, n_2) \tag{6.36}$$

where $\alpha_k$ and $\beta_k$ denote the $x_1$ and $x_2$ components of the displacement of frame $k$ with respect to the reference frame, $\delta(x_1, x_2)$ denotes the 2D Dirac delta function, and the low-resolution frames are assumed to be $N \times N$. Because of the linearity of convolution, Eqn. (6.36) can be equivalently expressed as

$$g_k(n_1, n_2) = \sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1} [s_0(x_1, x_2) ** h_c(x_1 - \alpha_k, x_2 - \beta_k)]\ \delta(x_1 - n_1\Delta, x_2 - n_2\Delta) + v_k(n_1, n_2) \tag{6.37}$$

Suppose we wish to reconstruct an $M \times M$ high-resolution sampled version of the reference frame $s_0(x_1, x_2)$, where $M$ is an integer multiple of $N$, i.e., $R=M/N$ is an integer. Assuming that $s_0(x_1, x_2)$ is bandlimited, such that

$$|S_0(F_1, F_2)| = 0 \quad \text{for } |F_1|, |F_2| > R\frac{1}{2\Delta}$$

the model (6.37) under-samples $s_0(x_1, x_2)$ by a factor $R$ in both $x_1$ and $x_2$ directions. Taking the Fourier transform of both sides of the model (6.37), we obtain

$$G_k(f_1, f_2)$$
$$= \sum_{i_1=0}^{R-1} \sum_{i_2=0}^{R-1} \frac{1}{\Delta^2} \left[ S_0\left(\frac{f_1 - i_1}{\Delta}, \frac{f_2 - i_2}{\Delta}\right) H_c\left(\frac{f_1 - i_1}{\Delta}, \frac{f_2 - i_2}{\Delta}\right) e^{-j\frac{2\pi}{\Delta}\{(f_1 - i_1)\alpha_k + (f_2 - i_2)\beta_k\}} \right]$$
$$+ V_k(f_1, f_2) \tag{6.38}$$

where $S_0\left(\frac{f_1}{\Delta}, \frac{f_2}{\Delta}\right)$ and $H_c\left(\frac{f_1}{\Delta}, \frac{f_2}{\Delta}\right)$ denote the 2D continuous Fourier transform of the reference frame $s_0(x_1, x_2)$ and the sensor PSF $h_c(x_1, x_2)$, respectively, and $G_k(f_1, f_2)$ and $V_k(f_1, f_2)$ denote the 2D discrete-Fourier transform of $g_k(n_1, n_2)$ and $v_k(n_1, n_2)$, respectively. We note that motion blur can be incorporated into the simplified model (6.38) (i.e., the assumption $T_A = 0$ may be relaxed) only if we have global, constant-velocity motion, so that the frequency response of the motion blur remains the same from frame-to-frame.

In order to recover the unaliased spectrum $S_0\left(\frac{f_1}{\Delta}, \frac{f_2}{\Delta}\right)$ at a given frequency pair $(f_1, f_2)$, we need at least $R^2$ equations (6.38) at that frequency pair, which could be obtained from $L > R^2$ low-resolution frames. The set of equations (6.38), $k = 1, \ldots, L$, at any frequency pair $(f_1, f_2)$ are decoupled from the equations that are formed at any other frequency pair. The formulation of Tsai and Huang [Tsa 84] ignores sensor blur and noise and proposes to set up $L > R^2$ equations in as many unknowns at each frequency pair to recover the alias-free spectrum $S_0\left(\frac{f_1}{\Delta}, \frac{f_2}{\Delta}\right)$. This procedure is illustrated by the following example. If we take the sensor blur into account, we can first recover the product $S_0\left(\frac{f_1}{\Delta}, \frac{f_2}{\Delta}\right) H_c\left(\frac{f_1}{\Delta}, \frac{f_2}{\Delta}\right)$. Subsequently, $S_0\left(\frac{f_1}{\Delta}, \frac{f_2}{\Delta}\right)$ can be estimated by inverse filtering or any other regularized de-convolution technique.

### Example: Frequency-Domain Super-Resolution: 1D Case

Consider two low-resolution observations of a 1D signal ($L = 2$), which are shifted with respect to each other by a sub-sample amount $\alpha$, such that

$$g_1(n) = \sum_{n=0}^{N-1} s_0(x)\,\delta(x - n\Delta) + v(n)$$
$$g_2(n) = \sum_{n=0}^{N-1} s_0(x - \alpha)\,\delta(x - n\Delta) + v(n)$$

where $N$ is the number of low-resolution samples and the sensor PSF is assumed to be a delta function.

Assuming that $g_k(n)$, $k=1,2$, are both sampled at one-half of the Nyquist rate ($R=2$), and taking the Fourier transform of both sides, we have

$$G_1(f)=\frac{1}{\Delta}S_0\left(\frac{f}{\Delta}\right)+\frac{1}{\Delta}S_0\left(\frac{f-1}{\Delta}\right)$$

$$G_2(f)=\frac{1}{\Delta}S_0\left(\frac{f}{\Delta}\right)e^{-j\frac{2\pi}{\Delta}f\alpha}+\frac{1}{\Delta}S_0\left(\frac{f-1}{\Delta}\right)e^{-j\frac{2\pi}{\Delta}(f-1)\alpha}$$

There is only one aliasing term in each expression, because we assumed sampling at half the Nyquist rate. We can solve for the two unknowns $S_0(f/\Delta)$ and $S_0((f-1)/\Delta)$ from these two equations given $G_1(f)$ and $G_2(f)$. Repeating this at each frequency sample, the spectrum of the alias-free HR signal can be recovered.

The following remarks about super-resolution in the frequency domain are in order:

1. The frequencies in the range $0\leq f_1, f_2\leq 1$ are discretized by $K$ samples along each direction, where $K\geq M$, and $M$ is the number of high-resolution signal samples. Then, samples of the high-resolution frame can be computed by a $K\times K$ inverse 2D-DFT.
2. It is easy to see that if the image $s_c(x_1, x_2, 0)$ were passed through a perfect anti-alias filter before the low-resolution sampling, there would be only one term in the double summation in Eqn. (6.38), and recovery of a high-resolution image would not be possible no matter how many low-resolution frames were available. Thus, it is the aliasing terms that make the recovery of a high-resolution image possible.

The frequency-domain approach has some drawbacks:

1. The set of equations (6.38) at each frequency pair $(f_1,f_2)$ can best be solved in the least-squares sense due to the presence of observation noise. Thus, more than $L^2$ equations, hence more frames, are needed to be solved for the $L^2$ unknowns at each frequency pair $(f_1, f_2)$.
2. The set of equations (6.38) may be singular depending on the relative position of the sub-pixel displacements, $\alpha_k$ and $\beta_k$, or due to zero-crossings in the Fourier transform $H_c(f_1,f_2)$ of the sensor PSF. In particular, if there are more than $L$ frames with shifts $(\alpha_k,\beta_k)$ on a line parallel to either the $x_1$ or $x_2$ axis, or if there are more than $L(L^2-L-1)/2$ pairs of frames with shifts $(\alpha_k,\beta_k)$ that

are symmetric with respect to the line $x_1 = x_2$, the system of equations becomes singular. This fact is stated by a theorem in [Kim 90]. Furthermore, it is clear from Eqn. (6.38) that any zeros in $H_c(f_1, f_2)$ result in a column of zeros in the coefficient matrix of the system of equations. Then, regularization techniques need to be employed that limit the resolution improvement.

3. This approach has been extended by Kim *et. al.* [Kim 90] to take noise and blur in the low-resolution images into account, where blur and noise characteristics need to be the same for all frames of the low-resolution data, and impulse sampling has been assumed for the low-resolution images (i.e., the low-resolution sensor has no physical size). This method was further refined by Kim and Su [Kim 93] to take into account blurs that are different for each frame of low-resolution data by using a Tikhonov regularization. The resulting algorithm does not treat the formation of blur due to motion or sensor size, and may suffer from convergence problems.

## 6.5.4 Multi-Frame Spatial-Domain Methods

The general SR-reconstruction problem formulated by the space-varying image formation model (6.35) can be addressed by spatial-domain methods that can be classified as early two-step interpolation-restoration methods, regularized SR-reconstruction methods, including Bayesian and set-theoretic methods, and methods that do not require sub-pixel motion estimation. They are reviewed in the following.

An important application of SR reconstruction is creating a single high-quality still image, a snapshot, from a video clip. Video snapshots [Sun 12] is a recent system that combines SR, de-noising, and deblurring with importance (saliency) weighting to produce either a snapshot that suppresses independently moving objects, or a snapshot that summarizes the motion of salient objects in a single frame.

### Interpolation-Restoration Methods

Early work on SR was based on two-stage interpolation-restoration methods, which essentially first map all pixels from available low-resolution frames onto a single up-sampled reference frame by using image-registration techniques. However, unless we assume global, constant-velocity motion, the up-sampled reference frame contains non-uniformly spaced samples. In order to obtain a uniformly spaced up-sampled image, an interpolation onto a uniform sampling grid needs to be performed. Then, a post-processing step, where image restoration is applied to the up-sampled image to remove the effect of the sensor PSF blur has been used. Sauer and Allebach [Sau 87] propose an iterative method to reconstruct band-limited images from non-uniformly

spaced samples. Their method estimates image intensities on a uniformly spaced grid using a projections-based method. Ur and Gross [Ur 92] have proposed a non-uniform interpolation scheme based on the generalized sampling theorem of Papoulis to obtain an improved-resolution blurred image, which is then restored using an image-restoration step. An important drawback of these algorithms is that they do not address removal of aliasing artifacts. Furthermore, the restoration stage neglects errors in the interpolation stage.

### Regularized-Reconstruction Methods

Super-resolution is an inverse problem. The forward process of modeling LR-image formation results in a set of simultaneous linear equations given by Eqn. (6.35). Hence, the solution of the set of simultaneous linear equations given by (6.35) is an inverse problem. Suppose that the desired high-resolution (HR) frame is $M \times M$, and we have $L$ LR frames, each $N \times N$. Then, we can set up $L \times N \times N$ equations in $M^2$ unknowns to reconstruct the HR frame. These equations are linearly independent provided that all displacements between LR frames are sub-pixel. Clearly, the number of equations will be reduced by the number of occluded pixels encountered along the motion trajectories. In general, it is desirable to set up an overdetermined system of equations, i.e., the number of LR frames $L > R^2 = M^2/N^2$, to obtain a robust solution. We note that fast methods to solve the set of simultaneous equations (6.35) are not available, because the impulse-response coefficients $h_k(n_1, n_2, m_1, m_2)$ are spatially varying in general; hence, the system matrix is not block-Toeplitz.

The inverse problem is ill-posed, because the presence of any noise in LR images and small errors in sub-pixel motion estimates leads to large errors in the solution. To this effect, regularization of the solution by incorporation of *a priori* image and noise models (similar to those used for image de-noising and restoration) is required to obtain a stable solution in the presence of noise. Hence, the problem is formulated as optimization of some regularization cost function subject to the constraint that the estimated high-resolution image is consistent with all observed low-resolution images and *a priori* models, which is solved iteratively.

All regularization methods that were discussed for solving the image-restoration problem (see Section 3.6) are also applicable to solving the SR-reconstruction problem. The regularization methods for solving the SR-reconstruction problem include i) constrained least squares [Par 03], ii) iterative back-projection [Ira 91, Ira 93], iii) set-theoretic methods including the projection onto convex sets (POCS) method [Tek 92, Pat 97b], iv) Bayesian methods including maximum likelihood (ML) and maximum a posteriori probability (MAP) [Sch 96], v) hybrid methods combining MAP and POCS [Ela 97], and vi) methods using self-similarity and sparsity models

[Tak 09]. We note that Komatsu *et al.* [Kom 93] have made the observation that using multiple cameras with different pixel apertures can overcome some limitations in the camera/motion configurations that may be encountered when using cameras with the same pixel aperture.

### *Bayesian Super-Resolution*

The MAP-estimation framework is a popular Bayesian approach to solve ill-posed image restoration and super-resolution reconstruction problems iteratively as a non-linear optimization problem. In this framework, the image-formation model is stated in a stochastic form by representing the model error

$$v_k(n_1,n_2) = g_k(n_1,n_2) - \sum_{m_1}\sum_{m_2} s(m_1,m_2)h_k(m_1,m_2;n_1,n_2)$$

as Gaussian observation noise. Schultz and Stevenson [Sch 96] employed discontinuity-preserving Huber–Markov Gibbs priors as the *a priori* image model. They formulated a constrained optimization problem with a unique minimum that can be found by iterative methods. More details can be found in [Sch 96, Par 03].

Excellent results have been reported for sequences with global frame-to-frame motion, such as camera pan or other global warping. More modest resolution improvements were observed for scenes with independent object motions [Sch 96].

### Set-Theoretic Methods

Set-theoretic methods address the general super-resolution problem in the POCS framework, including the special cases of multi-frame shift-varying restoration. In the following, we present a POCS-based method to solve a set of simultaneous linear equations (6.35) [Tek 92, Pat 97b]. A similar but more limited solution was also proposed by Stark and Oskoui [Sta 89]. The POCS formulation presented here is similar to that in Section 3.6.4 for intra-frame restoration of shift-varying blurred images.

We define a different closed, convex set for each observed low-resolution pixel $(n_1, n_2, k)$ that is connected to the desired high-resolution reference frame by a motion trajectory as follows:

$$C_{n_1,n_2,k} = \left\{x[m_1,m_2] \,\middle|\, \left|r_k^{\mathbf{x}}(n_1,n_2)\right| \leq \delta_0\right\},\ 0 \leq n_1,n_2 \leq N-1,\ k=1,\ldots,L \quad (6.39)$$

where

$$r_k^{\mathbf{x}}(n_1,n_2) = g_k[n_1,n_2] - \sum_{m_1=0}^{M-1}\sum_{m_2=0}^{M-1} x[m_1,m_2]h_k(m_1,m_2,n_1,n_2)$$

and $\delta_0$ represents the confidence we have in the observation, set equal to $c\,\sigma_v$, where $\sigma_v$ is the standard deviation of the noise and $c \geq 0$ is determined by an appropriate statistical confidence bound. These sets define high-resolution images that are consistent with the observed low-resolution frames within a confidence bound that is proportional to the variance of the observation noise.

The projection $y(m_1, m_2) \equiv \mathbf{P}_{n_1,n_2,k}\{x(m_1, m_2)\}$ of an arbitrary $x(m_1, m_2)$ onto $C_{n_1,n_2,k}$ is defined as

$$\mathbf{P}_{n_1,n_2,k}\{x(m_1,m_2)\} =$$
$$\begin{cases} x(m_1,m_2) + \dfrac{r_k^{\mathbf{x}}(n_1,n_2) - \delta_0}{\sum_o \sum_p h_k^2(o,p,n_1,n_2)} h_k(m_1,m_2,n_1,n_2) & \text{if } r_k^{\mathbf{x}}(n_1,n_2) > \delta_0 \\ x_i(m_1,m_2) & \text{if } -\delta_0 \leq r_k^{\mathbf{x}}(n_1,n_2) \leq \delta_0 \\ x(m_1,m_2) + \dfrac{r_k^{\mathbf{x}}(n_1,n_2) + \delta_0}{\sum_o \sum_p h_k^2(o,p,n_1,n_2)} h_k(m_1,m_2,n_1,n_2) & \text{if } r_k^{\mathbf{x}}(n_1,n_2) < -\delta_0 \end{cases} \quad (6.40)$$

Additional constraints, such as amplitude and/or finite support constraints, can be utilized to improve the results. The amplitude constraint $C_{\mathrm{A}}$ has been defined in (3.112), and the projection $\mathbf{P}_{\mathrm{A}}$ onto the amplitude constraint $C_{\mathrm{A}}$ is given by (3.113).

Given the above projection operators, an estimate $\hat{s}(m_1, m_2)$ of the high-resolution image $s(m_1, m_2)$ is obtained by successive projections onto each observation set iteratively as

$$\hat{s}^{(j+1)}[m_1, m_2] = \mathbf{T}_A \prod_{n_1=0}^{N-1} \prod_{n_2=0}^{N-1} \prod_{k=1}^{L^2} \mathbf{T}_{n_1,n_2,k} \left\{ \hat{s}^{(j)}[m_1, m_2] \right\}$$
$$0 \leq m_1,\, m_2 \leq M-1,\, j = 0,1,\ldots \quad (6.41)$$

where $\mathbf{T}$ denotes the generalized projection operator. An initial estimate $\hat{s}^{(0)}(m_1, m_2)$ of the high-resolution image is computed by interpolating the low-resolution reference frame to the desired resolution using bilinear or bicubic interpolation. The projections aim to ensure that the reconstructed high-resolution image is consistent with each and every pixel of all the observed low-resolution images as depicted in Figure 6.26. Excellent super-resolution reconstructions have been reported using this procedure [Tek 92, Pat 97b].

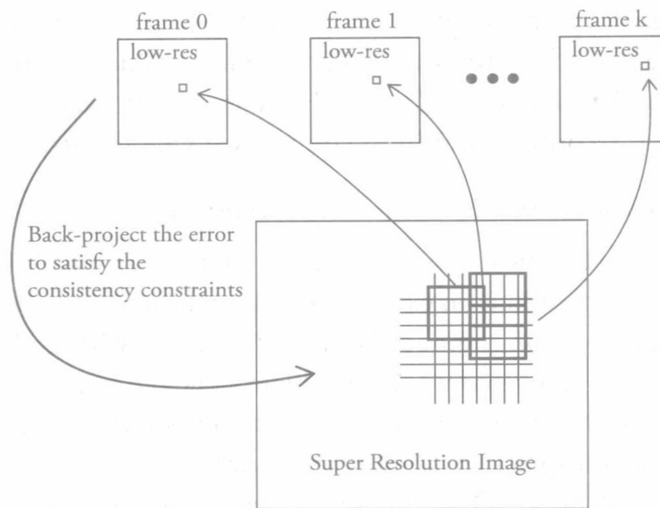



**Figure 6.26** Successive projections onto convex sets.

A few observations about the POCS method are in order:

1. While certain similarities exist between the POCS iterations and the Landweber-type iterations [Tru 85, Ira 91, Ira 93], the POCS method can adapt to the amount of observation noise, while the latter generally cannot.
2. The proposed POCS method can also be applied to shift-varying multi-frame restoration and standards conversion problems by specifying the input and output lattices and the shift-varying system PSF $h_k(m_1, m_2, n_1, n_2)$ appropriately (as stated in Section 3.6.4).
3. The POCS method finds a feasible solution, i.e., a solution consistent with all available low-resolution observations. Clearly, the more low-resolution observations (more frames with reliable sub-pixel motion estimates) are available, the better the high-resolution reconstructed image $\hat{s}(m_1, m_2)$ will be. In general, it is desirable that $L > M^2 / N^2$. Note, however that, the POCS method generates a reconstructed image with any number $L$ of available frames. The number $L$ is just an indicator of how large the feasible set of solutions will be. Note that the size of the feasible set can be further reduced by employing other closed, convex constraints in the form of statistical or structural image models.

There are several variations of the basic POCS solution to the super-resolution reconstruction problem. Elad and Feuer [Ela 97] proposed a hybrid super-resolution

reconstruction algorithm that combines the benefits of the MAP and POCS methods. The hybrid approach defines a single optimum solution while enforcing all convex constraints. Altunbasak *et al.* [Alt 02] incorporated quantization constraints for super-resolution reconstruction from compressed video bitstreams.

***Super-Resolution without Sub-Pixel Motion Estimation***

Since precise sub-pixel optical flow estimation in the presence of independently moving objects and occlusion is an exceedingly difficult problem, the practical applicability of SR-reconstruction methods is limited to video with global motion, e.g., affine camera motion. To overcome this problem, Takeda *et al.* [Tak 09] have proposed 3D steering kernel regression (in space-time) for super-resolution, which does not require explicit, sub-pixel accurate motion estimation. In this approach, each pixel is approximated by a 3D Taylor series, whose coefficients are estimated by solving a local weighted least-squares problem. The weights capture the 3D space-time orientation in the local neighborhood, which implicitly contains information about the local motion of the pixels across time, therefore rendering unnecessary an explicit computation of sub-pixel motion estimates. They have developed an iterative implementation of this algorithm with rough (pixel accurate) motion compensation to accommodate fast and complex motions.

## References

[Alt 02] Altunbasak, Y., A. J. Patti, and R. M. Mersereau, "Super-resolution still and video reconstruction from MPEG coded video," IEEE Trans. on Circuits and Systems for Video Technology, vol. 12, no. 4, pp. 217–227, April 2002.

[Arc 91] Arce, G. R., "Multistage order statistic filters for image sequence processing," IEEE Trans. Signal Proc., vol. 39, pp. 1146–1163, 1991.

[Bak 02] Baker, S., and T. Kanade, "Limits on super-resolution and how to break them," IEEE Trans. Patt. Anal. Mach. Intell., vol. 24, no. 9, pp. 1167–1183, Sept. 2002.

[Bar 10] Bartels, C., C. N. Cordes, B. Riemens, and G. De Haan, "A system approach to high-quality picture rate conversion," Jour. of the SID, vol. 18, no. 11, pp. 922–930, 2010.

[Bie 86] Bierling, M., and R. Thoma, "Motion compensating field interpolation using a hierarchically structured displacement estimator," Signal Processing, vol. 11, pp. 387–404, 1986.

[Bov 00] Bovik, Al, ed., *Handbook of Image and Video Processing,* San Diego, CA: Academic Press, 2000.

[Boy 92] Boyce, J., "Noise reduction of image sequences using adaptive motion compensated frame averaging," Proc. IEEE Int. Conf. Acoust. Speech and Sign. Proc., San Francisco, CA, pp. III-461–464, Mar. 1992.

[Caf 90] Cafforio, C., F. Rocca, and S. Tubaro, "Motion compensated image interpolation," IEEE Trans. Comm., vol. COM-38, pp. 215–222, Feb. 1990.

[Cap 90] Capodiferro, L., "Interlaced to progressive conversion by median filtering," Signal Proc. of HDTV II, L. Chiariglione, ed., Elsevier, 1990.

[Dav 78] Davis, L. S., and A. Rosenfeld, "Noise cleaning by iterated local averaging," IEEE Trans. Syst. Man and Cybern., vol. 8, pp. 705–710, 1978.

[Den 80] Dennis, T. J., "Non-linear temporal filter for television picture noise reduction," Proc. of IEE, vol. 127G, pp. 52–56, 1980.

[Dub 84] Dubois, E., and S. Sabri, "Noise reduction in image sequences using motion-compensated temporal filtering," IEEE Trans. Comm., vol. 32, pp. 826–831, July 1984.

[Dub 92] Dubois, E., "Motion-compensated filtering of time-varying images," Multidim. Syst. Sign. Proc., vol. 3, pp. 211–239, 1992.

[Ela 97] Elad, M., and A. Feuer, "Restoration of a single superresolution image from several blurred, noisy and undersampled measured images," IEEE Trans. on Image Proc., vol. 6, no. 12, pp. 1646–1658, Dec. 1997.

[Fre 02] Freeman, W. T., T. R. Jones, and E. C. Pasztor, "Example-based super-resolution," IEEE Computer Graphics and Applications, vol. 22, pp. 56–65, March/April 2002.

[Gir 85] Girod, B., and R. Thoma, "Motion-compensated field interpolation from interlaced and non-interlaced grids," SPIE Conf. Image Coding, pp. 186–193, 1985.

[Gir 93] Girod, B., "Motion compensation: Visual aspects, accuracy, and fundamental limits," in Motion Analysis and Image Sequence Processing, M. I. Sezan and R. L. Lagendijk, eds., Norwell, MA: Kluwer, 1993.

[Haa 92] Haavisto, P., and Y. Neuvo, "Motion adaptive scan rate up-conversion," Multidim. Syst. and Sign. Processing, vol. 3, pp. 113–130, 1992.

[Haa 93] de Haan, G., P. W. A. C. Biezen, H. Huijgen, and O. A. Ojo, "True motion estimation with 3D recursive search block matching," IEEE Trans. on Circ. and Syst. for Video Tech., vol. 3, no. 5, pp. 368–379, Oct. 1993.

[Haa 98] de Haan, G., and E. B. Bellers, "Deinterlacing – An overview," Proc. of the IEEE, vol. 86, no. 9, pp. 1839–1857, Sept. 1998.

[Hei 11] Heinrich, A., C. Bartels, R. J. van der Vleuten, C. N. Cordes, and G. de Haan, "Optimization of hierarchical 3DRS motion estimators for picture rate

conversion," IEEE J. of Selected Topics in Signal Processing, vol. 5, no. 2, pp. 262–274, Apr. 2011.

[Ira 91] Irani, M., and S. Peleg, "Improving resolution by image registration," CVGIP: Graphical Models and Image Proc., vol. 53, pp. 231–239, May 1991.

[Ira 93] Irani, M., and S. Peleg, "Motion analysis for image enhancement: Resolution, occlusion and transparency," J. Vis. Comm. and Image Rep., vol. 4, pp. 324–335, Dec. 1993.

[Kim 90] Kim, S. P., N. K. Bose, and H. M. Valenzuela, "Recursive reconstruction of high-resolution image from noisy undersampled frames," IEEE Trans. on Acoust., Speech and Sign. Proc., vol. 38, pp. 1013–1027, June 1990.

[Kim 93] Kim, S. P., and W.-Y. Su, "Recursive high-resolution reconstruction of blurred multiframe images," IEEE Trans. on Image Proc., vol. 2, pp. 534–539, Oct. 1993.

[Kom 93] Komatsu, T., T. Igarashi, K. Aizawa, and T. Saito, "Very high-resolution imaging scheme with multiple different aperture cameras," Signal Processing: Image Comm., vol. 5, pp. 511–526, Dec. 1993.

[Lag 92] Lagendijk, R. L., and M. I. Sezan, "Motion compensated frame rate conversion of motion pictures," Proc. of the IEEE ICASSP, San Francisco, CA, 1992.

[Lee 94] Lee, M., J. Kim, J. Lee, K. Ryu, and D. Song, "A new algorithm for interlaced to progressive scan conversion based on directional correlations and its IC design," IEEE Trans. on Consumer Electronics, vol. 40, pp. 119–129, May 1994.

[Lim 90] Lim, Jae S., *Two-Dimensional Signal and Image Processing*, Englewood Cliffs, NJ: Prentice Hall, 1990.

[Lin 04] Lin, Z., and H.-Y. Shum, "Fundamental limits of reconstruction-based superresolution algorithms under local translation," IEEE Trans. on Patt. Anal. Mach. Intell., vol. 26, no. 1, pp. 83–97, Jan. 2004.

[Liu 10] Liu, C., and W. T. Freeman, "A high-quality video denoising algorithm based on reliable motion estimation," Proc. of the ECCV, pp. 706–719, Springer-Verlag, 2010.

[Mag 12] Maggioni, M., G. Boracchi, A. Foi, and K. Egiazarian, "Video denoising, deblocking, and enhancement though separable 4-D nonlocal spatiotemporal transforms," IEEE Trans. on Image Proc., vol. 21, no. 9, pp. 3952–3966, Sept. 2012.

[Mar 85] Martinez, D., and J. S. Lim, "Implicit motion compensated noise reduction of motion video scenes," Proc. IEEE ICASSP, pp. 375–378, Tampa, FL, 1985.

[Ozk 92] Ozkan, M. K., A. T. Erdem, M. I. Sezan, and A. M. Tekalp, "Efficient multiframe Wiener restoration of blurred and noisy image sequences," IEEE Trans. on Image Proc., vol. 1, pp. 453–476, Oct. 1992.

[Ozk 93] Ozkan, M. K., M. I. Sezan, and A. M. Tekalp, "Adaptive motion-compensated filtering of noisy image sequences," IEEE Trans. on Circuits and Syst. for Video Tech., vol. 3, pp. 277–290, Aug. 1993.

[Par 03] Park, S. C., M. K. Park, and M. G. Kang, "Super-resolution image reconstruction: A technical overview," IEEE Signal Processing Magazine, pp. 21–36, May 2003.

[Pat 97a] Patti, A. J., M. I. Sezan, and A. M. Tekalp, "Robust methods for high-quality stills from video in the presence of dominant motion," IEEE Trans. on Circ. Syst. for Video Tech., vol. 7, no. 2, pp. 328–342, Apr. 1997.

[Pat 97b] Patti, A. J., M. I. Sezan, and A. M. Tekalp, "Super-resolution video reconstruction with arbitrary sampling lattices and non-zero aperture time," IEEE Trans. on Image Proc., vol. 6, no. 8, pp. 1064–1076, Aug. 1997.

[Pat 98] Patti, A. J., A. M. Tekalp, and M. I. Sezan, "A new motion-compensated reduced order model Kalman filter for restoration of progressive and interlaced video," IEEE Trans. on Image Processing, vol. 7, no. 4, pp. 543–554, April 1998.

[Sam 85] Samy, R., "An adaptive image sequence filtering scheme based on motion detection," SPIE, vol. 596, pp. 135–144, 1985.

[Sau 87] Sauer, K. D., and J. P. Allebach, "Iterative reconstruction of bandlimited images from non-uniformly spaced samples," IEEE Trans. on Circ. and Syst., vol. 34, pp. 1497–1505, 1987.

[Sch 87] Schamel, G., "Pre- and post-filtering of HDTV signals for sampling rate reduction and display up-conversion," IEEE Trans. Circuits and Syst., vol. 34, pp. 1432–1439, Nov. 1987.

[Sch 96] Schultz, R. R., and R. L. Stevenson, "Extraction of high-resolution frames from video sequences," IEEE Trans on Image Processing, vol. 5, no. 6, pp. 996–1011, June 1996.

[Sez 91] Sezan, M. I., M. K. Ozkan, and S. V. Fogel, "Temporally adaptive filtering of noisy image sequences using a robust motion estimation algorithm," Proc. IEEE Int. Conf. Acoust. Speech, Signal Processing, pp. 2429–2432, Toronto, Canada, 1991.

[Sta 89] Stark, H., and P. Oskoui, "High-resolution image recovery from image plane arrays using convex projections," J. Opt. Soc. Amer. A, vol. 6, pp. 1715–1726, 1989.

[Sun 12] Sunkavalli, K., N. Joshi, S. B. Kang, M. F. Cohen, and H. Pfister, "Video snapshots: Creating high-quality images from video clips," IEEE Trans. on Visualization and Computer Graphics, vol.18, no. 11, pp. 1868–1879, Nov. 2012.

[Tak 09] Takeda, H., P. Milanfar, M. Protter, and M. Elad, "Super-resolution without explicit subpixel motion estimation," IEEE Trans. on Image Processing, vol. 18, no. 9, pp. 1958–1975, Sept. 2009.

[Tek 92] Tekalp, A. M., M. K. Ozkan, and M. I. Sezan, "High-resolution image reconstruction from lower-resolution image sequences and space-varying image restoration," IEEE ICASSP, San Francisco, CA, vol. III, pp. 169–172, March 1992.

[Tho 89] Thoma, R., and M. Bierling, "Motion compensating interpolation considering covered and uncovered background," Signal Processing: Image Comm., vol. 1, pp. 191–212, 1989.

[Tru 85] Trussell, H. J., and M. Civanlar, "The Landweber iteration and projection onto convex sets," IEEE Trans. on Acoust. Speech Sign. Proc., vol. 33, pp. 1632–1634, Dec. 1985.

[Tru 92] Trussell, H. J., and S. Fogel, "Identification and restoration of spatially variant motion blurs in sequential images," IEEE Trans. on Image Proc., vol. 1, pp. 123–126, Jan. 1992.

[Tub 93] Tubaro, S., and F. Rocca, "Motion field estimators and their application to image interpolation," in Motion Analysis and Image Sequence Processing, M. I. Sezan and R. L. Lagendijk, eds., Norwell, MA: Kluwer, 1993.

[Tsa 84] Tsai, R. Y., and T. S. Huang, "Multiframe image restoration and registration," in Advances in Computer Vision and Image Processing, vol. 1, T. S. Huang, ed., pp. 317–339, Greenwich, CT: JAI Press, 1984.

[Uns 90] Unser, M., and M. Eden, "Weighted averaging of a set of noisy images for maximum signal-to-noise ratio," IEEE Trans. on Acoust., Speech and Sign. Proc., vol. 38, pp. 890–895, May 1990.

[Ur 92] Ur, H., and D. Gross, "Improved resolution from subpixel shifted pictures," CVGIP: Graphical Models and Image Processing, vol. 54, pp. 181–186, March 1992.

[Wan 90] Wang, F.-M., D. Anastassiou, and A. N. Netravali, "Time-recursive de-interlacing for IDTV and pyramid coding," Signal Proc.: Image Comm., vol. 2, pp. 365–374, 1990.

[Woo 91] Woods, J. W., and S.-C. Han, "Hierarchical motion-compensated de-interlacing," SPIE Vis. Comm. Image Proc., vol. 1605, pp. 805–810, 1991.

[Yam 94] Yamauchi, T., N. Ouchi, and H. Shimano, "Motion-compensated TV standards converter using motion vectors computed by an iterative gradient method," Signal Proc.: Image Comm., vol. 6, pp. 267–274, 1994.

[Zac 93] Zaccarin, A., and B. Liu, "Block motion compensated coding of interlaced sequences using adaptively de-interlaced fields," Signal Proc.: Image Comm., vol. 5, pp. 473–485, 1993.

# Exercises

## Problem Set 6

**6.1** Let the horizontal and vertical bandwidths of a video signal with an unknown global constant velocity be $10^6$ cyc/mm. Suppose it is sampled on a vertically aligned 2:1 interlaced lattice with the parameters $\Delta_{x_1} = \Delta_{x_2} = 100$ microns, and $\Delta_t = 1/60$sec. Find all critical velocities, if any.

**6.2** Show that the spatio-temporal impulse response of the filter that performs "merging" of even and odd fields to form a composite frame is given by

$$h(x_2, t) = \delta(x_2)\,\delta(t) + \delta(x_2)\,\delta(t + T)$$

where $T$ is the field interval. Find the frequency response of this filter. Discuss frequency-domain interpretation of merging for stationary and moving image regions.

**6.3** Compare the relative advantages and disadvantages of two-, three-, and four-frame motion-detection algorithms from interlaced video.

**6.4** How would you deal with motion-estimation errors in motion-compensated up-conversion? How would you compare motion-adaptive filtering and adaptive motion-compensated filtering in the presence of motion-estimation errors?

**6.5** How would you compare the motion-compensated adaptive LMMSE and AWA filters in the presence of a sudden scene change?

**6.6** Discuss the frequency response of the MCMF filter (Section 6.4) as it relates to the theory of motion-compensated filtering presented in Section 6.1.

**6.7** Consider constant-velocity global motion, where the velocity is modeled as constant during each aperture time (piecewise constant-velocity model). Let its value during $i$th aperture time (acquisition of the $i$th frame) be given by $\mathbf{v}_i = [v_{1i}\ v_{2i}]^T$. Show that the Jacobian $J(u_1, u_2, t)$ in Eqn. (6.31) is equal to $1/|v_{1i}|$

**6.8** Suppose we have four low-resolution images that have global translations with respect to a reference frame with velocities $v_{1i} = v_{2i} = 8.25$, 8.5, 9, and 10 in units of pixels per high-resolution sampling grid for $i = 1, \ldots, 4$, respectively. Assume that each side of the low-resolution sensor cell is four times that of the high-resolution cell with rectangular pixel geometry shown in Figure 6.25. Calculate the discrete-discrete PSF $h_{ik}(m_1, m_2, n_1, n_2)$ in Eqn. (6.35).

**6.9** Derive Eqn. (6.38).

**6.10** Derive Eqn. (6.40).

**6.11** Discuss the relationship between the POCS reconstruction discussed in Section 6.5.4 and the back-projection iterations presented in [Ira 93].

## MATLAB Exercises

**6.1** De-interlacing

Suppose we have a standard-definition interlaced color video stored in a .yuv file in composite-frame format, where pixels for each frame are ordered as all Y pixels first, followed by Cb pixels and Cr pixels sequentially. Note that the chrominance components are 4:2:0 sub-sampled; hence, the Y component is 704×480 pixels, and Cb and Cr are each 352×240 pixels.

**a.** Extract (separate) the even and odd fields given an interlaced video .yuv file.

**b.** Implement the bob filter (intra-averaging) to generate full-size frames from even and odd fields.

**c.** Implement the weave filter to generate full size frames from even and odd fields.

**d.** Implement the motion-adaptive bob-and-weave filter to generate full-size frames from even and odd fields.

Compare results and write your observations about the results.

**6.2** AWA Filter

Given a video sequence with $N$ frames

**a.** Add zero-mean, white Gaussian noise with variance $\sigma_v^2$ to each frame.

**b.** Estimate motion trajectory at each pixel. Use any motion-estimation method of your choice. Comment on the quality of the motion estimates as the variance of the noise increases.

**c.** Implement the AWA filter along the motion trajectories given by (6.25). How do you select $a>0$ and $\varepsilon$? Comment on the effect of these parameters on the performance of the filter as a function of the noise variance $\sigma_v^2$.

CHAPTER 7

# Image Compression

*Compression may be mathematically lossless or lossy. The more the visual-quality degradation (loss) that can be tolerated, the higher the compression ratio will be. Compression of images without significant loss of perceived quality is possible because images contain a high degree of i) spatial redundancy, due to correlation between neighboring pixels, ii) spectral redundancy, due to correlation among color components, and iii) psychovisual redundancy, due to what the human eye cannot see. The more the redundancy, the higher the achievable compression will be.*

The need for effective data compression is evident in almost all applications where storage and transmission of digital images are involved. For example, an 8.5×11 document scanned at 300 pixels/in with 1 bit/pixel generates 8.4 Mbits data. A high-resolution digital still image with 4000 pixels × 3000 lines and 8 bits/pixel per color is 288 Mbits. This chapter introduces the basics of image compression, and discusses some commonly used lossless and lossy still-image-compression methods and standards. Further references are included for those who wish to implement the covered compression algorithms and standards. Some preliminary concepts on information theory as well as elements of an image-compression system, including quantization and entropy coding, are introduced in Section 7.1. Most popular

lossy image-coding algorithms employ the transform-coding para-digm. Section 7.2 covers discretecosine transform (DCT) based image coding and the International Standards Organization (ISO) JPEG standard for lossy image compression. Wavelet-transform based lossless and lossy image coding and the ISO JPEG2000 standard are discussed in Section 7.3.

## 7.1 Basics of Image Compression

This section first summarizes some basic results from information theory that provide bounds on the achievable compression ratios and bit-rates, then presents the basic elements of a general image-compression system.

### 7.1.1 Information Theoretic Concepts

A source $\boldsymbol{X}$ with an alphabet $\boldsymbol{A}$ is defined as a discrete random process (a sequence of random variables $X_i$, $i=1, \ldots$) in the form $\boldsymbol{X}=X_1 X_2, \ldots$, where each random variable $X_i$ takes a value from the alphabet $\boldsymbol{A}$. In the following, we assume the alphabet contains $M$, a finite number of symbols, i.e., $\boldsymbol{A}=\{a_1, a_2, \ldots, a_M\}$. A discrete memoryless source (DMS) is such that successive symbols are statistically independent. It is completely specified by the probabilities $p(a_i)=p_i$, $i=1, \ldots, M$ such that $p_1+\ldots+p_M=1$.

According to information theory, the information content of a symbol is related to the extent that the symbol is unpredictable or unexpected. If a symbol with low probability occurs, a larger amount of information is transferred than in the occurrence of a more likely symbol. This quantitative concept of surprise is formally expressed by the relation

$$I(a_i) = \log_2\left(\frac{1}{p(a_i)}\right), \text{ for } a_i \in \boldsymbol{A} \tag{7.1}$$

where $I(a_i)$ is the information that the symbol $a_i$ with probability $p(a_i)$ carries. The unit of information is bits when we use logarithm with base-2. Observe that if $p=1$, then as expected $I=0$, and at the other extreme, $I \rightarrow \infty$ as $p \rightarrow 0$.

In variable-length source coding (VLC), in the case where we assign an individual codeword to each individual symbol, the optimum length of the binary code for a symbol is equal to the information (in bits) of the symbol. In practice, the probability of occurrence of each symbol is estimated from the histogram of a given source or a training set of sources.

The entropy $H(\boldsymbol{X})$ of a DMS $\boldsymbol{X}$ with an alphabet $\boldsymbol{A}$ is defined as the average information per symbol in the source, given by

$$H(\boldsymbol{X}) = \sum_{a_i \in \boldsymbol{A}} p(a_i) \log_2\left(\frac{1}{p(a_i)}\right) = -\sum_{a_i \in \boldsymbol{A}} p(a_i) \log_2\left(p(a_i)\right) \qquad (7.2)$$

The more skewed the probability distribution of the symbols, the smaller the entropy of the source. The entropy is maximized for a flat distribution, i.e., when all symbols are equally likely. It follows that a source where some symbols are more likely than others has a smaller entropy than another source where all symbols are equally likely. Hence, the performance of lossless encoding of a source will be related to the entropy of the source.

### Example: Entropy of a Raw Image

Suppose an 8-bit image is taken as a realization of a DMS $\boldsymbol{X}$. The symbols $i$ are gray levels of pixels in the image, and the alphabet $\boldsymbol{A}$ is the set of all possible gray levels between 0 and 255. Then, the entropy of the image is given by

$$H(\boldsymbol{X}) = -\sum_{i=0}^{255} p(i) \log_2\left(p(i)\right)$$

where $p(i)$ denotes the relative frequency of occurrence of the gray level $i$ in the image. Note that the entropy of an image consisting of a single gray level (constant image) is zero.

Next, we present two fundamental theorems, the lossless-coding theorem and the source-coding theorem, which are used to assess the performance of lossless coding and lossy coding methods, respectively.

**Lossless-Coding Theorem [Sha 48]**

The minimum bit-rate that can be achieved by lossless coding of a DMS $\boldsymbol{X}$ is given by

$$R_{\min} = H(\boldsymbol{X}) + \varepsilon \text{ bits/symbol}$$

where $R$ is the transmission rate, $H(\boldsymbol{X})$ is the entropy of the source, and $\varepsilon$ is a positive quantity that can be made arbitrarily close to zero.

The lossless-coding theorem establishes the lower bound for the bit-rate necessary to achieve zero coding-decoding error in the case of a DMS. We will introduce

Huffman coding, which can approach this bound for DMS by encoding each symbol independently, and arithmetic coding, which assigns a single codeword to an arbitrary length group of input symbols.

In lossy coding, the achievable minimum bit-rate is a function of the distortion that is allowed. This relationship between the bit-rate and distortion is given by the rate-distortion function [Ber 71] as stated by the source-coding theorem.

### Source-Coding Theorem

There exists a mapping from the source symbols to codewords such that for a given distortion $D$, $R(D)$ bits/symbol are sufficient to enable source reconstruction with an average distortion that is arbitrarily close to $D$. The actual rate $R$ should obey

$$R \geq R(D)$$

for fidelity $D$. The function $R(D)$ is called the rate-distortion function. Note that $R(0) = H(\boldsymbol{X})$.

A typical rate-distortion function is depicted in Figure 7.1. The rate-distortion function can be computed analytically for simple source and distortion models. Computer algorithms exist to compute $R(D)$ when analytical methods fail or are unpractical [Ber 71]. In general, we are interested in designing a compression system to achieve either the lowest bit-rate for a given distortion or the lowest distortion at a given bit-rate. Note that the source-coding theorem does not state how to design algorithms to achieve these desired limits.

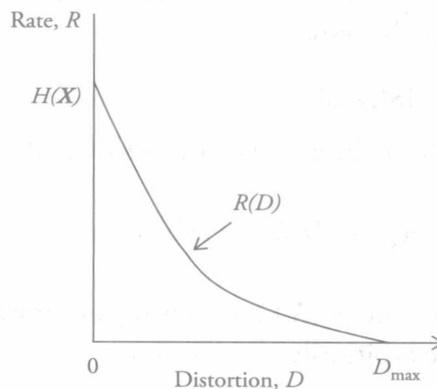



**Figure 7.1** Rate-distortion function.

## 7.1.2 Elements of Image-Compression Systems

In information theory, the process of data compression by redundancy reduction is referred to as source encoding. Images contain two types of redundancy: statistical (spatial) and pyschovisual. Statistical redundancy is present because certain spatial patterns are more likely than others, whereas psychovisual redundancy originates from the fact that the human eye is insensitive to certain spatial frequencies. The block diagram of a source encoder is shown in Figure 7.2. It is composed of the following blocks:

1. *Transformer* (T) applies a one-to-one transformation to the input image data. The output of the transformer is an image representation that is more amenable to efficient compression than the raw image data. Typical transformations are linear predictive mapping, which maps the pixel intensities onto a prediction error signal by subtracting the predictible part of the pixel intensities; unitary mappings such as the discrete-cosine transform, which pack the energy of the signal to a small number of coefficients; and multi-resolution mappings, such as sub-band decompositions and the wavelet transform.
2. *Quantizer* (Q) generates a limited number of symbols that can be used in the represention of the compressed image. Quantization is a many-to-one mapping that is irreversible.
3. *Coder* (C) assigns a binary codeword to each symbol at the output of the quantizer. It may employ fixed-length or variable-length codes.

Different image-compression systems implement different combinations of these choices. Image-compression methods can be broadly classified as:

- Lossless (noiseless) compression methods, which aim to minimize the bit-rate without any distortion in the image.
- Lossy compression methods, which aim to obtain the best possible fidelity for a given bit-rate, or to minimize the bit-rate to achieve a given fidelity measure.

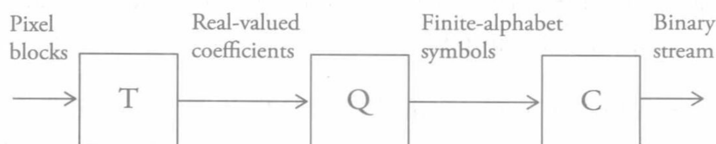



**Figure 7.2** Block diagram of a lossy image-compression system. A lossless compression system employs an integer transform and no quantization.

The transformation and encoding blocks are lossless. However, quantization is lossy. Therefore, lossless methods, which only make use of the statistical redundancies, do not employ a quantizer. In most practical cases, a small degradation in the image quality must be allowed to achieve the desired bit-rate. Lossy compression methods make use of both the statistical and psychovisual redundancies. In the following, we discuss quantization and entropy coding.

### 7.1.3 Quantization

Quantization is the process of representing a source consisting of set of continuous-valued samples with a finite number of states (also called a finite alphabet). It can be performed using scalar or vector quantizers. If each sample is quantized independently, the process is known as scalar quantization. Vector quantization refers to quantization of a block of samples, represented by a vector, at once, with a finite number of vector states [Ger 92]. Here, we focus on scalar quantizers since state-of-the-art image and video compression employs scalar quantization.

A scalar quantizer $Q(\cdot)$ is a function that is defined in terms of a finite set of decision levels $d_i$ and reconstruction levels $r_i$. The quantized variable, $\dot{s}$, is given by

$$\dot{s} = Q(s) = r_i \text{ if } s \in \left(d_{i-1}, d_i\right],\ i = 1, \ldots, L \tag{7.3}$$

where $L$ is the number of output quantized states. That is, the output of the quantizer is the reconstruction level $r_i$, if $s$, the value of the sample before quantization, is within the range $(d_{i-1}, d_i]$. The distance between successive decision (reconstruction) levels can be equal or unequal, called uniform and non-uniform quantization, respectively.

The performance of a quantizer is measured by a distortion measure $D$, which is a function of the quantization error, $e = s - \dot{s}$, that depends on $d_i$ and $r_i$. If we treat $s$ as a realization of a random variable $S$ with a probability density function (pdf) $p_S(s)$, the distortion measure may be taken as the mean-square quantization error $D = E\{(s - \dot{s})^2\}$, where $E\{\cdot\}$ stands for the expectation operator.

Given a distortion measure $D$ and the source pdf $p_s(s)$, there are two optimum scalar quantizer design methodologies:

1. For a fixed number of levels $L$, find $r_i$ and $d_i$, $i=0, \ldots, L$, in order to minimize distortion $D$. Non-uniform quantizers with a fixed number of levels that are optimal in the mean-square-error sense, $D = E\{(s - r_i)^2\}$, are known as Lloyd–Max quantizers [Llo 82, Max 60].

2. For a fixed output entropy $H(\cdot)=C$, where $C$ is a constant, find $r_i$ and $d_i$, $i=0,\ldots,L$, ($L$ is unknown) in order to minimize $D$. Quantizers that minimize a distortion measure for a constant output entropy are known as entropy-constrained quantizers [Woo 69].

A detailed discussion of the design of Lloyd-Max quantizers and entropy-constrained quantizers can be found in [Gra 98]. Image/video compression standards employ uniform quantization with a fixed number of levels, which is a special case of Lloyd-Max quantizers. This case is discussed in the following.

### Uniform Quantization

A quantizer is called a uniform quantizer if the distances between successive reconstruction levels are equal, i.e.,

$$r_{i+1} - r_i = \theta, \quad 1 \leq i \leq L-1$$

where $\theta$ is a constant called the step-size. Uniform quantizers can be classified as mid-tread or mid-riser. A mid-tread quantizer, described by

$$Q(s) = \mathrm{sgn}(s)\left\lfloor \frac{|s|}{\theta} + \frac{1}{2} \right\rfloor \theta = NINT\left(\frac{s}{\theta}\right)\theta \tag{7.4a}$$

where $\lfloor x \rfloor$ denotes the smallest integer greater than $x$ and $NINT$ denotes the nearest integer round-off, has a zero-valued reconstruction level, while a mid-riser quantizer described by

$$Q(s) = \left( \left\lfloor \frac{s}{\theta} \right\rfloor + \frac{1}{2} \right) \theta \tag{7.4b}$$

has a zero-valued decision level. Mid-tread quantizers should be preferred when the pdf of the source is symmetric about $s=0$ and decay for larger values of $s$. In mid-tread quantizers, the decision region around $s=0$ is called the deadzone.

#### Example: Uniform Quantization in JPEG Image Compression

In JPEG image compression, the pdf of the source (DCT coefficients) is modeled by a zero-mean Laplacian distribution, given by

$$p_S(s) = \frac{1}{\sigma\sqrt{2}} e^{-\frac{\sqrt{2}}{\sigma}|s|}$$

Hence, mid-tread uniform quantization given by (7.4a) is employed. The encoder computes the integer quantization index values

$$k = NINT\left(\frac{s}{\theta}\right)$$

that are entropy encoded for transmission/storage. The reconstructed values

$$\hat{s} = Q(s) = k\theta$$

are computed by the decoder, given the step-size $\theta$. JPEG employs human visual system weighted quantization, where a different step-size is used for each frequency coefficient position. The recommended (default) step-sizes have been determined by the JPEG committee.

### Example: Uniform Quantization of a Source Uniformly Distributed in [A,B]

If $p(s)$ is uniformly distributed over an interval $[A, B]$, then the uniform quantizer is the Lloyd–Max quantizer, and for $L$ levels, the step-size is given by

$$\theta = \frac{B - A}{L}$$

Then, using a mid-riser quantizer, the reconstruction levels are given by (7.4b).

### Example: Quantization Noise

Suppose we have a memoryless *zero*-mean Gaussian source $S$ with variance $\sigma^2$. Let the distortion measure be the mean-square error. What is the minimum number of levels, equivalently the rate $R$ in bits/sample, to obtain 40 dB SNR, assuming uniform quantization? We can express the mean-square quantization noise as

$$D = E\left\{(s - \hat{s})^2\right\}$$

Then, the SNR in dB is given by

$$\text{SNR} = 10 \log_{10} \frac{\sigma^2}{D}$$

SNR=40 dB implies $\frac{\sigma^2}{D} = 10{,}000$. Substituting this into the rate-distortion function for a memoryless Gaussian source, given by

$$R(D) = \frac{1}{2} \log_2 \frac{\sigma^2}{D}$$

we can compute $R(D) \approx 7$ bits/sample. Similarly, we can show that quantization with 8 bits/sample yields approximately 48 dB SNR.

### 7.1.4 Symbol Coding

Symbol coding is the process of assigning a binary string to individual symbols or to a block of symbols comprising the source. We start by discussing the simplest scheme, which is to assign equal-length codewords to individual symbols or a fixed-length block of symbols, known as fixed-length coding. Often, significantly better compression can be achieved by assigning shorter-length codewords to more probable symbols, which is the main principal of entropy coding. As a result, most image- and video-compression schemes employ entropy coding.

#### Fixed-Length Coding

Fixed-length coding assigns equal-length codewords to each symbol in the alphabet $\boldsymbol{A}$ regardless of their probabilities. If the alphabet has $M$ different symbols (or blocks of symbols), then the length of the codes is the smallest integer greater than $\log_2 M$. Two commonly used fixed-length coding schemes are natural codes and Gray codes, which are shown in Table 7.1 for the case of a four-symbol source. Notice that in Gray coding, the consecutive codewords differ in only one bit position. This property may provide an advantage in error detection. Gray codes are also better suited for run-length encoding of bit-planes.

**Table 7.1** Fixed-Length Codes for a Four-Symbol Alphabet

| Symbol | Natural Code | Gray Code |
|---|---|---|
| $a_1$ | 00 | 00 |
| $a_2$ | 01 | 01 |
| $a_3$ | 10 | 11 |
| $a_4$ | 11 | 10 |

It can easily be shown that fixed-length coding is optimal only when: i) the number of symbols is equal to a power of 2, and ii) all the symbols are equiprobable. Only then will the entropy of the source be equal to the average length of the codewords, which is equal to the length of each codeword in the case of fixed-length coding. For the example shown in Table 7.1, both the entropy of the source and the average codeword length is 2, assuming all symbols are equally likely.

**Entropy Coding**

In most compression applications, some symbols are more probable than others, where it would be more advantageous to use entropy coding, which assigns variable-length codewords to each symbol. Entropy coding, also known as variable-length coding (VLC), assigns codewords in such a way as to minimize the average codeword length for the source. This is achieved by assigning shorter codewords to more probable symbols, which is the fundamental principle of entropy coding. Indeed, the goal of the transformation box in Figure 7.2 is to obtain a set of symbols with a skew probability distribution to minimize the entropy of the transformed source.

Two popular methods of entropy coding are Huffman coding and arithmetic coding, which are introduced in more detail in the next two sub-sections. The first, Huffman coding, assigns variable-length codes to a fixed-length block of symbols, where the block length is typically one, and the length of the codewords is proportional to the information (in bits) of the respective symbols or block of symbols. The latter, arithmetic coding, assigns variable-length codes to a variable-length block of symbols.

Note that the rate $R$ to encode the quantized sample values in the case of fixed-length coding is given by $R = \lfloor \log_2 L \rfloor$, where $\lfloor x \rfloor$ denotes the smallest integer greater than $x$, while in the case of entropy coding or VLC, it is given by $R > H(\hat{s})$ according to the lossless coding theorem.

## 7.1.5 Huffman Coding

Huffman coding yields the optimal integer prefix codes given a source with a finite number of symbols and their probabilities. In prefix codes, no codeword is a prefix of another codeword. Such codes are uniquely decodable since a given binary string can only be interpreted in one way. Huffman codes are optimal in the sense that no other integer-length VLC can be found to yield a smaller average bit-rate. In fact, the average length of Huffman codes per codeword achieves the lower bound, the entropy of the source, when the symbol probabilities are all powers of 2. Huffman codes can be designed by following a very simple procedure.

Let $\boldsymbol{X}$ denote a DMS with the alphabet $\boldsymbol{A}$ and the symbol probabilities $p(a_i)$, $a_i \in \boldsymbol{A}$, $i=1, \ldots, M$. Obviously, if $M=2$, we must have

$$c(a_1) = 0 \text{ and } c(a_2) = 1 \tag{7.5}$$

where $c(a_i)$ denotes the codeword for the symbol $a_i$, i=1, 2. If $\boldsymbol{A}$ has more than two symbols, the Huffman procedure requires a series of source reduction steps. In each step, we find and merge the two symbols with the smallest probabilities, which results in a new source with a reduced alphabet. The probability of the new symbol in the reduced alphabet is the sum of the probabilities of the two "merged" symbols from the previous alphabet. This procedure is continued until we reach a source with only two symbols, for which the codeword assignment is given by Eqn. (7.5). Then we work backwards toward the original source, each time splitting the codeword of the "merged" symbol into two new codewords by appending it with a zero and one, respectively. The following examples demonstrate this procedure.

### Example: Symbol Probabilities are Powers of 2

Let the alphabet $\boldsymbol{A}$ consist of four symbols, shown in Table 7.2. The probabilities and information of the symbols in the alphabet are also listed in the table. Note that all symbol probabilities are powers of 2, and consequently the symbols have integer information values.

The Huffman-coding procedure for this alphabet is given in Table 7.3 and Figure 7.3. The reduced alphabet in step 1 is obtained by merging the symbols $a_3$ and $a_4$ in the original alphabet that have the lowest two probabilities. Likewise, the reduced alphabet in step 2 is obtained by merging the two symbols with the lowest probabilities after step 1. Since the reduced alphabet in step 2 has only two symbols, we assign the codes 0 and 1 to these symbols in arbitrary order. Next, we assign codes to the reduced alphabet in step 1. Recall that the symbol 2 in step 2 is obtained by merging the symbols 2 and

**Table 7.2** An Alphabet Where the Symbol Probabilities Are Powers of 2

| Symbol | Probability | Information |
|---|---|---|
| $a_1$ | 0.5 | 1 |
| $a_2$ | 0.25 | 2 |
| $a_3$ | 0.125 | 3 |
| $a_4$ | 0.125 | 3 |

Table 7.3 Illustration of Alphabet Reduction

| Original Alphabet | | Reduced Alphabet Step 1 | | Reduced Alphabet Step 2 | |
|---|---|---|---|---|---|
| $p$ | $c$ | $p$ | $c$ | $p$ | $c$ |
| 0.5 | 0 | 0.5 | 0 | 0.5 | 0 |
| 0.25 | 10 | 0.25 | 10 | 0.5 | 1 |
| 0.125 | 110 | 0.25 | 11 | | |
| 0.125 | 111 | | | | |

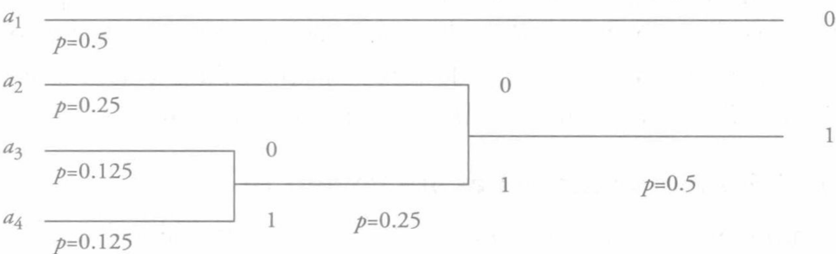



Figure 7.3 Tree diagram for Huffman coding.

3 in step 1. Thus, we assign codes to symbols 2 and 3 in step 1 by appending the code for symbol 2 in step 2 by a zero and one in arbitrary order. The appended zero and one are shown in bold fonts in Table 7.3. Finally, the codes for the original alphabet are obtained in a similar fashion. This procedure can alternatively be described by the tree diagram shown in Figure 7.3.

Observe that in this case, the average codeword length is

$$\bar{R} = 0.5 \times 1 + 0.25 \times 2 + 0.125 \times 3 + 0.125 \times 3 = 1.75$$

and the entropy of the source is

$$H = -0.5 \log_2 0.5 - 0.25 \log_2 0.25 - 0.125 \log_2 0.125 - 0.125 \log_2 0.125 = 1.75$$

which is consistent with the result that Huffman coding achieves the entropy of the source when the symbol probabilities are powers of 2. Next, we present an example in which the symbol probabilities are not powers of 2.

## Example: Symbol Probabilities are Not Powers of 2

When the probabilities of the symbols are not powers of 2, the information content of each symbol is a real number, as shown in Table 7.4.

Since the length of each codeword must be an integer, it is not possible to design codewords whose lengths are equal to the information of the respective symbols in this case. Huffman-code design for the alphabet in Table 7.4 is shown in Table 7.5. It can be easily seen that for this example the average length of codewords is 2.15 and entropy of the source is 2.07.

Notice that Huffman codes are uniquely decodable, with proper synchronization, because no codeword is a prefix of another. For example, a received binary string

$$001101101110000\ldots$$

can be decoded uniquely as

$$a_3\ a_1\ a_2\ a_1\ a_2\ a_1\ a_1\ a_4\ \ldots$$

**Table 7.4** An Alphabet Where the Symbol Probabilities Are Not Powers of 2

| Symbol | Probability | Information |
|---|---|---|
| $a_1$ | 0.4 | 1.32 |
| $a_2$ | 0.25 | 2 |
| $a_3$ | 0.15 | 2.73 |
| $a_4$ | 0.15 | 2.73 |
| $a_5$ | 0.05 | 4.32 |

**Table 7.5** Huffman Coding When Probabilities Are Not Powers of 2

| Original Alphabet | | Reduced Alphabet Step1 | | Reduced Alphabet Step2 | | Reduced Alphabet Step 3 | |
|---|---|---|---|---|---|---|---|
| $p$ | $c$ | $p$ | $c$ | $p$ | $c$ | $p$ | $c$ |
| 0.4 | 1 | 0.4 | 1 | 0.4 | 1 | 0.6 | **0** |
| 0.25 | 01 | 0.25 | 01 | 0.35 | 0**0** | 0.4 | **1** |
| 0.15 | 001 | 0.20 | 00**0** | 0.25 | 0**1** | | |
| 0.15 | 000**0** | 0.15 | 00**1** | | | | |
| 0.005 | 000**1** | | | | | | |

**Block Coding**

Until now, we have discussed scalar coding, where each symbol in the alphabet is assigned an individual code. Block coding refers to the case where we do not assign a separate code to each symbol, but assign codewords to blocks of $L$ symbols from the original alphabet. Of course, this requires building a new block alphabet with all possible combinations of the $L$ symbols from the original alphabet and computing their respective probabilities. Huffman codes for all possible combinations of the $L$ symbols from the original alphabet can be formed using the previously described design procedure with the new block alphabet. Thus, Huffman coding may be considered a block-coding scheme, where we assign variable-length codes to fixed-length ($L$) blocks of symbols. The case $L=1$ refers to assigning an individual codeword to each symbol of the original alphabet, as shown in the previous examples. It has been shown that for sources with memory, the coding efficiency improves as $L$ gets larger, although the design of the Huffman codes gets more complicated.

## 7.1.6 Arithmetic Coding

In arithmetic coding, a one-to-one correspondence between the symbols of an alphabet $\boldsymbol{A}$ and the codewords does not exist. Instead, arithmetic coding assigns a single variable-length code to a source $\boldsymbol{X}$, composed of $N$ symbols from the alphabet, where $N$ is variable. The distinction between arithmetic coding and block Huffman coding is that in arithmetic coding the length of the input sequence, i.e., the block of symbols for which a single codeword is assigned, is variable. Thus, arithmetic coding assigns variable-length codewords to variable-length blocks of symbols. Because arithmetic coding does not require assignment of integer-length codes to fixed-length blocks of symbols, in theory it can asymptotically achieve the lower bound established by the lossless-coding theorem.

Arithmetic coding associates a given realization of $\boldsymbol{X}$, $\mathbf{x}=\{x_1,\ldots,x_N\}$, with a sub-interval of $[0,1)$ whose length equals the probability of the sequence $p(\mathbf{x})$. The encoder processes the input stream of symbols one by one, starting with $N=1$, where the length of the sub-interval associated with the sequence gets smaller as $N$ increases. Bits are sequentially sent to the channel starting from the most-significant bit toward the least-significant bit as they are determined according to the procedure presented in an algorithmic form in the following. At the end of the transmission, the transmitted bitstream is a uniquely decodable codeword representing the source, which is a binary number pointing to the sub-interval associated with this sequence.

### Procedure

Consider an alphabet $\boldsymbol{A}$ with $M$ symbols $a_i$, $i=1,\ldots,M$, with probabilities $p(a_i)=p_i$, such that $p_1+\ldots+p_M=1$. We start by assigning each individual symbol in the alphabet a sub-interval, within 0 to 1, whose length is equal to its probability. It is assumed that this assignment is known to the decoder.

1. If the first input symbol $x_1=a_i, i=1,\ldots,M$, then define the initial sub-interval as $\mathrm{I}_1=[l_1,r_1)=[p_{i-1},p_{i-1}+p_i)$, where $p_0=0$. Set $n=1$, $L=l_1$, $R=r_1$, and $d=r_1-l_1$.
2. Obtain the binary expansions of $L$ and $R$ as

$$L=\sum_{k=1}^{\infty} u_k 2^{-k} \text{ and } R=\sum_{k=1}^{\infty} v_k 2^{-k} \tag{7.6}$$

   where $u_k$ and $v_k$ are 0 or 1. Compare $u_1$ and $v_1$.
   a. If they are not the same, send nothing to the channel, and go to step 3.
   b. If $u_1=v_1$, then send the binary symbol $u_1$, and compare $u_2$ and $v_2$. If they are not the same, go to step 3.
   c. If $u_2=v_2$, also send the binary symbol $u_2$, and compare $u_3$ and $v_3$, and so on, until the next two corresponding binary symbols do not match, at which time go to step 3.
3. Increment $n$, and read the next symbol. If the $n$th input symbol $x_n=a_i$, then sub-divide the interval from the previous step as

$$\mathrm{I}_n=[l_n,r_n)=[l_{n-1}+p_{i-1}d,l_{n-1}+(p_{i-1}+p_i)d)$$

   Set $L=l_n$, $R=r_n$, and $d=r_n-l_n$, and go to step 2.

Note that the decoder may decode one binary symbol into several source symbols, or it may require several binary symbols before it can decode one or more source symbols. The operations of the encoder and decoder are illustrated by the following example.

#### Example: Arithmetic Coding

Let's determine an arithmetic code to represent a sequence of symbols,

$$a_2\, a_1\, a_3 \ldots$$

from the source shown in Table 7.3. Because we have four symbols in the alphabet, the interval between 0 and 1 is initially sub-divided into four,

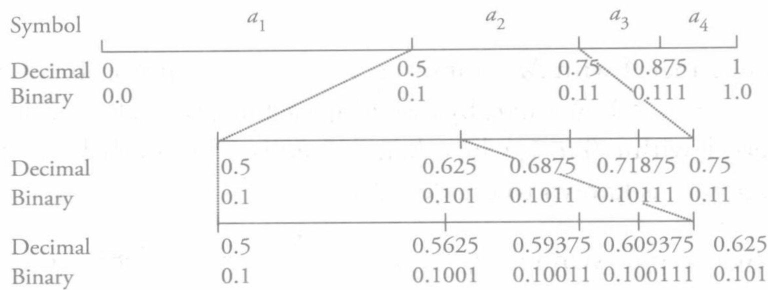



**Figure 7.4** Illustration of the concept of arithmetic coding.

where the lengths of the sub-intervals are equal to 0.5, 0.25, 0.125, and 0.125, respectively. This is depicted in Figure 7.4.

The first symbol defines the initial interval as $I_1=[0.5, 0.75)$, where the binary representations of the left and right boundaries are $L=2^{-1}=0.1$ and $R=2^{-1}+2^{-2}=0.11$, respectively. According to step 2, $u_1=v_1=1$; thus, "1" is sent to the channel. Noting that $u_2=0$ and $v_2=1$, we read the second symbol, $a_1$. Step 3 indicates that $I_2=[0.5,0.625)$, with $L=0.10$ and $R=0.101$. Now that $u_2=v_2=0$, we send "0" to the channel. However, $u_3=0$ and $v_3=1$, so we read the third symbol, $a_3$. It can be easily seen that $I_3=[0.59375, 0.609375)$, with $L=0.10011$ and $R=0.100111$. Note that $u_3=v_3=0$, $u_4=v_4=1$, and $u_5=v_5=1$, but $u_6=0$ and $v_6=1$. At this stage, we send "011" to the channel, and read the next symbol. A reserved symbol usually signals the end of a sequence.

Let's now briefly look at how the decoder operates, which is illustrated in Table 7.6. The first bit restricts the interval to [0.5,1). However, three symbols are within this range; thus, the first bit does not contain sufficient information. After receiving the second bit, we have "10," which points to the interval [0.5,0.75). All possible combinations of two symbols pointing to this range start with $a_2$. Hence, we can now decode the first symbol as $a_2$. The information that becomes available after the receipt of each bit is summarized in Table 7.6.

Although we assumed fixed-symbol probabilities above, arithmetic coding allows adapting probability tables after encoding each symbol, which is called *adaptive arithmetic coding* [Pen 88]. The updates must be computed in a "causal" manner that can be duplicated by the decoder so the encoder and decoder remain in synch at all times.

**Table 7.6** Operation of the Decoder

| Received Bit | Interval | Decoded Symbol |
|---|---|---|
| 1 | [0.5, 1) | – |
| 0 | [0.5, 0.75) | $a_2$ |
| 0 | [0.5, 0.625) | $a_1$ |
| 1 | [0.5625, 0.625) | – |
| 1 | [0.59375, 0.625) | – |
| ... | | |

In practice, two factors cause the performance of the arithmetic encoder to fall short of the theoretical bound: the use of finite-precision arithmetic and the addition of an end-of-message indicator. Practical implementations of the arithmetic coder overcome the precision problem by using a scaling and a rounding strategy [Wit 87].

## 7.2 Discrete-Cosine Transform Coding and JPEG

Transform coding, developed more than three decades ago, has proven to be an effective image-compression scheme, and is the basis of most world standards for lossy compression to date. A basic transform coder segments the image into small blocks. Each block undergoes a 2D orthogonal transformation to produce an array of transform coefficients. Next, the transform coefficients are quantized and coded. The coefficients having the highest energy over all blocks are most finely quantized, and those with the least energy are quantized coarsely or simply truncated. The encoder treats the quantized coefficients as symbols that are then entropy (variable-length) coded. The decoder reconstructs the pixel intensities from the received bitstream following the inverse operations on a block-by-block basis. The block diagrams of a transform encoder and decoder are shown in Figure 7.5.

The best transformation for the purposes of effective quantization should produce uncorrelated coefficients, and pack the maximum amount of energy (variance) into the smallest number of coefficients. The first property justifies the use of scalar quantization. The second property is desirable because we would like to discard as many coefficients as possible without seriously affecting image quality. The transformation that satisfies both properties is the Karhunen–Loeve transformation (KLT). Despite its favorable theoretical properties, the KLT is not used in practice, because: i) its basis functions depend on the covariance matrix of the image, hence they have to be recomputed and transmitted for every image, ii) perfect decorrelation is not

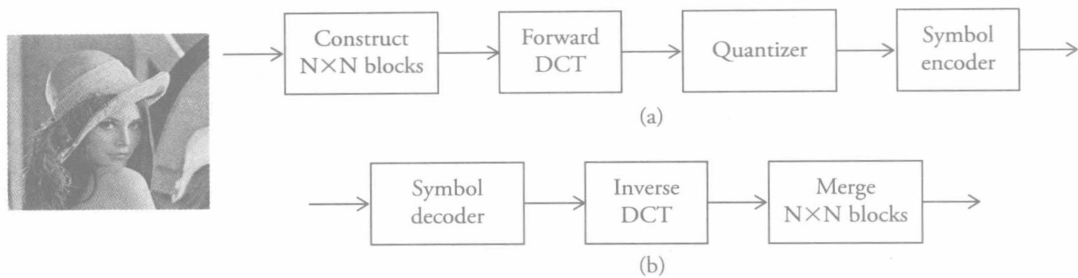



**Figure 7.5** Block diagram for transform coding: (a) encoder and (b) decoder.

possible, since images can rarely be modeled as realizations of homogeneous random fields, and iii) there are no fast algorithms for its implementation. After many considerations, the discrete-cosine transform (DCT), an orthonormal transform with data-independent basis functions, has been found to be the most effective with a performance close to that of the KLT [Net 88, Rab 91].

## 7.2.1 Discrete-Cosine Transform

The DCT is the most widely used transformation in transform coding. It is an orthogonal transform that has a fixed set of (image-independent) basis functions, an efficient algorithm for its computation, and good energy compaction and correlation-reduction properties [Rao 90]. Ahmed *et al.* [Ahm 74] first noticed that the KLT basis functions of a first-order Markov image closely resemble those of the DCT. They become identical as the correlation between adjacent pixels approaches one.

The DCT belongs to the family of discrete-trigonometric transforms, which has 16 members [Mar 94]. The type-2 DCT of an $N \times N$ block is defined as

$$S(k_1,k_2) = \sqrt{\frac{4}{N^2}}\; C(k_1)C(k_2) \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} s(n_1,n_2) \cos\left(\frac{\pi(2n_1+1)k_1}{2N}\right) \cos\left(\frac{\pi(2n_2+1)k_2}{2N}\right)$$

where $k_1$, $k_2$, $n_1$, $n_2$=0, 1, ..., $N-1$, and

$$C(k) = \begin{cases} 1/\sqrt{2} & \text{for } k=0 \\ 1 & \text{otherwise} \end{cases}$$

A significant factor in transform coding is the block size. The most popular sizes are 8×8 and 16×16, both powers of 2 for computational reasons. It is no surprise that the DCT is closely related to the DFT. In particular, an $N \times N$ DCT of $s(n_1, n_2)$ can be expressed in terms of a $2N \times 2N$ DFT of its even-symmetric extension, which

leads to a fast computational algorithm (see Section 1.2.4).

Using the separability of the DFT and one of several fast Fourier transform (FFT) algorithms, it is possible to compute an $N \times N$ DCT using $O(2N^2 \log_2 N)$ operations instead of $O(N^4)$, where $O(\cdot)$ stands for "order of." In addition, because of the even-symmetric extension process, no artificial discontinuities are introduced at the block boundaries, unlike the DFT, resulting in superior energy compaction. The fact that the computation of the DCT requires only real arithmetic facilitates its hardware implementation. As a result, DCT is widely available in special-purpose single-chip VLSI hardware, which makes it attractive for real-time use.

The energy-compaction property of the 8×8 DCT is illustrated in the following example.

### Example: Energy-Compaction Property of DCT

An 8×8 block of pixels from the 7th frame of the Mobile and Calendar sequence is:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 183 | 160 | 94 | 153 | 194 | 163 | 132 | 165 |
| 183 | 153 | 116 | 176 | 187 | 166 | 130 | 169 |
| 179 | 168 | 171 | 182 | 179 | 170 | 131 | 167 |
| 177 | 177 | 179 | 177 | 179 | 165 | 131 | 167 |
| 178 | 178 | 179 | 176 | 182 | 164 | 130 | 171 |
| 171 | 180 | 180 | 179 | 183 | 169 | 132 | 169 |
| 179 | 179 | 180 | 182 | 183 | 170 | 129 | 173 |
| 180 | 179 | 181 | 179 | 181 | 170 | 130 | 169 |

The DCT coefficients (nearest integer), after subtracting 128 from each pixel intensity, are:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 313 | 56 | −27 | 18 | 78 | −60 | 27 | −27 |
| −38 | −27 | 13 | 44 | 32 | −1 | −24 | −10 |
| −20 | −17 | 10 | 33 | 21 | −6 | −16 | −9 |
| −10 | −8 | 9 | 17 | 9 | −10 | −13 | 1 |
| −6 | 1 | 6 | 4 | −3 | −7 | −5 | 5 |
| 2 | 3 | 0 | −3 | −7 | −4 | 0 | 3 |
| 4 | 4 | −1 | −2 | −9 | 0 | 2 | 4 |
| 3 | 1 | 0 | −4 | −2 | −1 | 3 | 1 |

Observe that the high-frequency coefficients (around the lower-right corner) are much smaller than the low-frequency coefficients around (0,0) (at the upper-left corner).

### 7.2.2 ISO JPEG Standard

The JPEG standard describes a family of image-compression techniques for continuous-tone (gray-scale or color) still images. JPEG-baseline algorithm features lossy compression based on transform coding to remove statistical and pyschovisual redundancy. Work toward the JPEG standard got started in March 1986. In January 1988, JPEG reached consensus that the adaptive DCT approach should be the basis for the standard. The JPEG committee successfully finalized the international standard in July 1992 [Wal 91, Pen 93]. The JPEG standard supports:

1. Resolution independence: Arbitrary source resolutions can be handled. Images whose dimensions are not multiples of 8 are internally padded to multiples of 8 in DCT-based modes of operation.
2. Precision: DCT modes of operation are restricted to 8 and 12 bits/sample precision. For lossless coding the precision can be from 2 to 16 bits/sample, although JBIG has been found to perform better below 4 to 5 bits/sample.
3. No absolute bit-rate targets: The bit-rate/quality tradeoff is controlled primarily by the quantization matrix (see Section 7.2.3).
4. Luminance-chrominance separability: The ability exists to recover a luminance-only image from luminance-chrominance encoded images without always having to decode chrominance.

JPEG datastreams are defined in terms of what a JPEG decoder needs in order to decompress the datastream. No particular file format, spatial resolution, or color space model is specified as part of the standard. However, JPEG includes a minimal recommended file format, JPEG File Interchange Format (JFIF), which enables JPEG bit-streams to be exchanged between a wide variety of platforms and applications. In addition, several commonly available image-file formats are JPEG-compatible. In other words, the viewer or the application programs must recognize the specific file format in addition to being able to decode JPEG-compressed images.

JPEG provides four modes of operation: sequential (baseline), hierarchical, progressive, and lossless. The first three are discussed below. A JPEG- compatible product must support at least the baseline mode.

### Baseline Algorithm

The JPEG baseline mode is inspired by the scene adaptive coder [Che 84]. The main steps of the baseline algorithm can be summarized as:

1. DCT computation: The image is first sub-divided into 8×8 blocks. Each pixel is level-shifted by subtracting $2^{n-1}$, where $2^n$ is the maximum number of gray levels. That is, for 8-bit images we subtract 128 from each pixel in an attempt to remove the DC level of each block. The 2D-DCT of each block is then computed. In the baseline system, the input and output data precision is limited to 8 bits, whereas the quantized DCT values are restricted to 11 bits.
2. Quantization of the DCT coefficients: The DCT coefficients are threshold-coded using a quantization matrix and then reordered using zigzag scanning to form a 1D sequence of quantized coefficients. The quantization matrix can be scaled to provide a variety of compression levels. The entries of the quantization matrix are usually determined according to psychovisual considerations, which are discussed below.
3. Variable-length code (VLC) assignment: The non-zero AC coefficients are Huffman-coded using a VLC code that defines the value of the coefficient and the number of preceding zeros. Standard VLC tables are specified. The DC coefficient of each block is differential pulse-code modulation (DPCM)-coded relative to the DC coefficient of the previous block.

### Color

JPEG uses a standard color space (ITU-R 601-1). It transforms RGB images into a luminance-chrominance space, known as the Y-Cr-Cb space, defined by

$$Y = 0.3\,R + 0.6\,G + 0.1\,B$$

$$Cr = \frac{B-Y}{2} + 0.5$$

$$Cb = \frac{R-Y}{1.6} + 0.5$$

Because the human eye is relatively insensitive to the high-frequency content of the chrominance channels Cr and Cb (see Figure 3.2(b)), they are sub-sampled by 2 in both directions. Tis is illustrated in Figure 7.6, where the chrominance channels contain half as many lines and pixels per line compared to the luminance channel.

JPEG orders the pixels of a color image as either non-interleaved (three separate

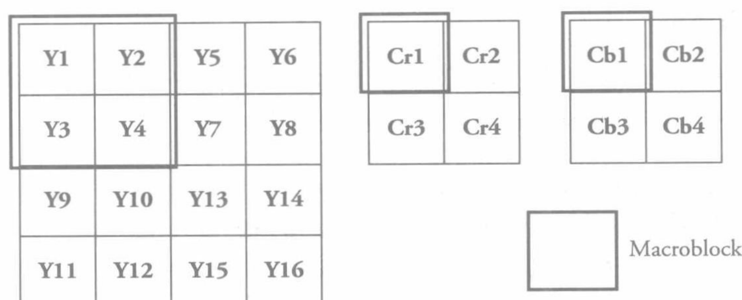



**Figure 7.6** Macroblock formation.

scans) or interleaved (a single scan). Referring to Figure 7.6, the non-interleaved ordering is given by

Scan 1: Y1, Y2, Y3, ..., Y16

Scan 2: Cr1, Cr2, Cr3, Cr4

Scan 3: Cb1, Cb2, Cb3, Cb4

whereas the interleaved ordering becomes

Y1, Y2, Y3, Y4, Cr1, Cb1, Y5, Y6, Y7, Y8, Cr2, Cb2, ...

Interleaving makes it possible to decompress the image and convert from luminance-chrominance representation to RGB for display with a minimum of intermediate buffering. For interleaved data, the DCT blocks are ordered according to the parameters specified in the frame and scan headers.

### Psychovisual Aspects

In order to exploit the psychovisual redundancy, JPEG incorporates characteristics of the human visual system through specification of quantization matrices. It is well known that the frequency response of the human visual system drops off with increasing spatial frequency. Furthermore, this drop-off is faster in chrominance channels, which is demonstrated by the contrast-sensitivity function depicted in Figure 2.4(b). It shows that small variations in intensity are more visible in slowly varying regions than in busier ones, and they are also more visible in the luminance components compared to the chrominance components.

To this effect, JPEG allows specification of two quantization matrices, one for the

luminance and another for the two chrominance channels, to allocate more bits to the representation of coefcients, which are visually more signifcant. Tables 7.7(a) and (b) show JPEG-specified default quantization matrices for the luminance and chrominance channels, respectively. The elements of these matrices are based on the visibility of individual 8×8 DCT basis functions with a viewing distance equal to six times the screen width. The basis functions are viewed with a luminance resolution of 720 pixels × 576 lines and a chrominance resolution of 360×576. The matrices suggest that DCT coefficients corresponding to basis images with low visibility can be more coarsely quantized.

### Example: Demonstration of the JPEG Baseline Algorithm

The JPEG baseline algorithm is applied to the 8×8 luminance block given in Section 7.2.1. Te gray levels are frst shifted by −128 (assuming that the original is an 8-bit image) and then a forward DCT is applied. The DCT

**Table 7.7** (a) Quantization Table for the Luminance Channel and (b) Quantization Table for the Chrominance Channel

(a)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 16 | 11 | 10 | 16 | 24 | 40 | 51 | 61 |
| 12 | 12 | 14 | 19 | 26 | 58 | 60 | 55 |
| 14 | 13 | 16 | 24 | 40 | 57 | 69 | 56 |
| 14 | 17 | 22 | 29 | 51 | 87 | 80 | 62 |
| 18 | 22 | 37 | 56 | 68 | 109 | 103 | 77 |
| 24 | 35 | 55 | 64 | 81 | 104 | 113 | 92 |
| 49 | 64 | 78 | 87 | 103 | 121 | 120 | 101 |
| 72 | 92 | 95 | 98 | 112 | 100 | 103 | 99 |

(b)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 17 | 18 | 24 | 47 | 99 | 99 | 99 | 99 |
| 18 | 21 | 26 | 66 | 99 | 99 | 99 | 99 |
| 24 | 26 | 56 | 99 | 99 | 99 | 99 | 99 |
| 47 | 66 | 99 | 99 | 99 | 99 | 99 | 99 |
| 99 | 99 | 99 | 99 | 99 | 99 | 99 | 99 |
| 99 | 99 | 99 | 99 | 99 | 99 | 99 | 99 |
| 99 | 99 | 99 | 99 | 99 | 99 | 99 | 99 |
| 99 | 99 | 99 | 99 | 99 | 99 | 99 | 99 |

coefcients divided by the quantization matrix, shown in Table 7.7(a), have been found as

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 20 | 5 | −3 | 1 | 3 | −2 | 1 | 0 |
| −3 | −2 | 1 | 2 | 1 | 0 | 0 | 0 |
| −1 | −1 | 1 | 1 | 1 | 0 | 0 | 0 |
| −1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Following a zigzag scanning (depicted in Figure 7.7) of these coefcients, the 1D coefficient sequence can be expressed as

$$20, 5, -3, -1, -2, -3, 1, 1, -1, -1, 0, 0, 1, 2, 3,$$
$$-2, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, \text{EOB}$$

where EOB denotes the end of the block (i.e., all following coefficients are zero.)

Te DC coefcients are coded using DPCM, as depicted in Figure 7.8. That is, the difference between the DC coefficient of the present block and that of the previously encoded block is coded. (Assume the DC coefficient of the previous block was 29, so the difference is −9.) The AC coefficients are mapped into symbols that are in the form of (RUN, LEVEL) pairs, given by

$$(0, 5), (0, -3), (0, -1), (0, -2), (0, -3), (0, 1), (0, 1), (0, -1), (0, -1),$$
$$(2, 1), (0, 2), (0, 3), (0, -2), (0, 1), (0, 1), (6, 1), (0, 1), (1, 1), \text{EOB}$$

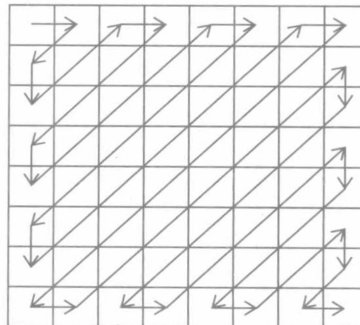


**Figure 7.7** Zigzag scan order.

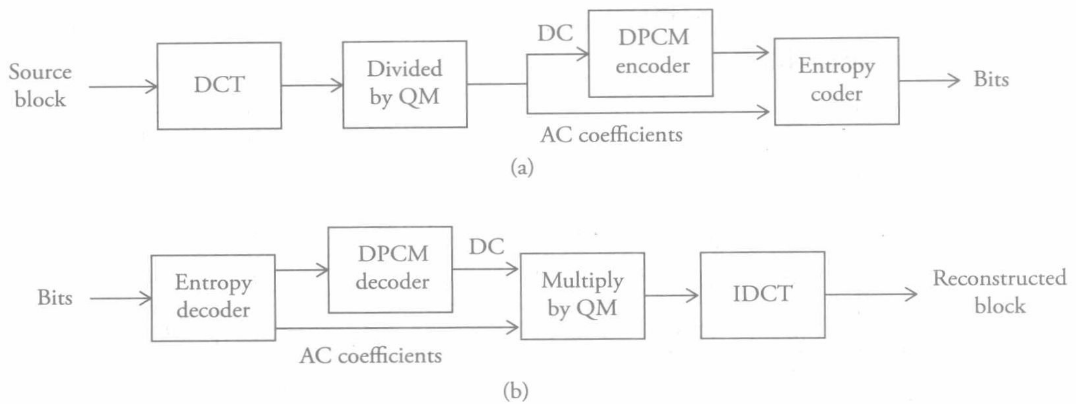



**Figure 7.8** Operation of JPEG baseline (a) encoder and (b) decoder.

where RUN indicates the number of zeros preceeding the non-zero coefficient value LEVEL in the zigzag-scanned 1D coefficient sequence.

The codewords for these symbols can be found according to the tables: JPEG coefficient coding categories, JPEG default DC codes (luminance/chrominance), and JPEG default AC codes (luminance/chrominance) [Gon 07, Pen 93]. For example, the DC difference −9 falls within the DC difference category 4. The proper base (default) code for category 4 is 011 (a 3-bit code), while the total length of a completely encoded category 4 coefficient is 7 bits. The remaining 4 bits will be the least-significant bits (LSBs) of the difference value. Each default AC Huffman codeword depends on the number of zero-valued coefficients preceding the non-zero coefficient to be coded as well as the magnitude category of the coefficient.

The decoder implements the inverse operations. That is, the received coefficients are first multiplied by the same quantization matrix to obtain

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 320 | 55 | −30 | 16 | 72 | −80 | 51 | 0 |
| −36 | −24 | 14 | 38 | 26 | 0 | 0 | 0 |
| −14 | −13 | 16 | 24 | 40 | 0 | 0 | 0 |
| −14 | 0 | 0 | 29 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Performing an inverse DCT, and adding 128 to each element, we find the reconstructed block

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 195 | 140 | 119 | 148 | 197 | 171 | 120 | 170 |
| 186 | 153 | 143 | 158 | 193 | 168 | 124 | 175 |
| 174 | 169 | 172 | 168 | 187 | 166 | 128 | 177 |
| 169 | 180 | 187 | 171 | 185 | 170 | 131 | 170 |
| 172 | 182 | 186 | 168 | 186 | 175 | 131 | 161 |
| 177 | 180 | 181 | 168 | 189 | 175 | 129 | 161 |
| 181 | 178 | 181 | 174 | 191 | 169 | 128 | 173 |
| 183 | 178 | 184 | 181 | 192 | 162 | 127 | 185 |

The reconstruction error is as large as $\pm 25$ gray levels, as can be seen by comparing the result with the original block shown in Section 7.2.1. Te compression ratio, and hence the quality of the reconstructed image, is controlled by scaling the quantization matrix (see Section 7.2.3).

### JPEG Progressive

The progressive DCT mode refers to encoding of the DCT coefficients in multiple passes, where a portion of the quantized DCT coefficients is transmitted in each pass. Two complementary procedures, corresponding to different groupings of the DCT coefficients, have been defined for progressive transmission: the spectral selection and successive approximation. An organization of the DCT coefficients for progressive transmission is depicted in Figure 7.9, where MSB and LSB denote the most-significant bit and the least-significant bit, respectively.

In the spectral selection process, the DCT coefficients are ordered into spectral bands where the lower-frequency bands are encoded and sent first. For example, the DC coefficient of each block is sent in the initial transmission, yielding a rather blocky first image at the receiver. The image quality is usually acceptable after the first five AC coefficients are also transmitted. When all the DCT coefficients are eventually coded and transmitted, the image quality is the same as that of the sequential algorithm.

In the successive approximation method, the DCT coefficients are first sent with lower precision, and then refined in successive passes. The DC coefficient of each block is sent first with full precision to avoid mean level mismatch. The AC coefficients may be transmitted starting with the MSB plane. Successive approximation

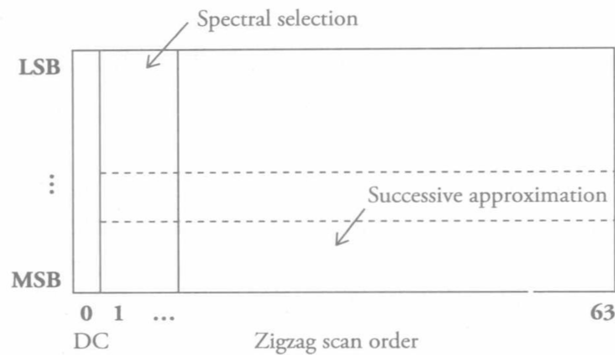



**Figure 7.9** Arrangement of the DCT coefficients for progressive transmission.

usually gives better-quality images at lower bit-rates. The two procedures may be intermixed by using spectral selection within successive approximations.

### JPEG Hierarchical

The hierarchical mode of operation employs the concept of pyramid coding [Bur 83] and may be considered as a special case of the progressive transmission, with increasing spatial resolution between the progressive stages. Multi-resolution image representation, in the form of the Gaussian pyramids depicted in Figure 3.12, was discussed in Section 3.2.3.

In the first stage, the lowest-resolution image (top layer of the Gaussian pyramid) is encoded using one of the sequential or progressive JPEG modes. The decoded output of each stage is then interpolated to form the prediction for the next stage. The bilinear interpolation filter that doubles the horizontal and vertical resolution is specified in the standard. In the next stage, the difference between the actual second layer in the Gaussian pyramid and the up-sampled first-layer image is encoded and transmitted. The procedure continues until the residual of the highest-resolution image is encoded and transmitted. In the hierarchical mode of operation, the image quality at extremely low bit-rates surpasses any of the other JPEG modes, but this is achieved at the expense of a higher bit rate at the completion of the progression.

## 7.2.3 Encoder Control and Compression Artifacts

Since JPEG compression quantizes DCT coefficients, there is always a loss of fidelity when an image is encoded and then decoded. If sufficiently fine quantization is used, the decoded pictures will be visually lossless (pixel differences will not be visible by humans) but the bit-rate (bits/pixel) or filesize will be larger. On the other hand,

by humans) but the bit-rate (bits/pixel) or filesize will be larger. On the other hand, if coarser quantization is employed then the encoding/decoding error will manifest itself as visible compression artifacts. These artifacts are called ringing and blocking (see Section 2.5.1).

In JPEG, the tradeoff between decoded image quality and bit-rate (filesize) can be controlled by scaling the default quantization matrix, which is given in Table 7.7. Scaling the quantization matrix by a factor larger than 1 results in a coarser quantization and a lower bit-rate at the expense of higher compression error. The compression ratio (CR) is defined as the ratio bits/pixel of the original image vs. the compressed image. For an 8-bit monochrome image, CR=8 (i.e., 1 bit/pixel encoding) results in visually lossless compression for most images. Severe ringing and blocking is observed at CR=15 or higher.

Most JPEG implementations give the user a parameter to trade image size for image quality, such as the quality factor (QF) parameter, which scales the quantization matrix. The range of values for QF is typically between 1–100, where QF=75 corresponds to the unscaled default matrix, but this may vary from implementation to implementation.

## 7.3 Wavelet-Transform Coding and JPEG 2000

Wavelet-transform coding is a multi-resolution image-coding approach that is closely related to earlier sub-band coding, which improves upon pyramid coding used in the hierarchical mode of the original JPEG. Both wavelet and sub-band coding employ a "complete" multi-resolution image representation, where the number of samples is equal to that in the original image, whereas pyramid coding uses an "overcomplete" pyramid-image representation with a larger number of samples than that of the original image.

The basic idea of wavelet/sub-band coding is to decompose an image into non-overlapping frequency bands to obtain a set of low-pass, bandpass, and high-pass sub-images. Each sub-image is then sub-sampled, encoded separately using a bit-rate matched to its perceptual requirements. The decoder reconstructs the image by adding upsampled and appropriately filtered versions of the decoded sub-images.

In the following, we first address the decomposition of an image into sub-images, and its reconstruction from these sub-images, which is called the analysis-synthesis problem. We also compare the filters used in sub-band and wavelet transforms. Specific methods used for coding the individual sub-images in sub-band coding vs. the JPEG2000 standard are presented next.

## 7.3.1 Wavelet Transform and Choice of Filters

The discrete-wavelet transform (DWT) is a multi-resolution image decomposition, which can be considered as expansion of an image onto a set of wavelet-basis functions. The wavelet-basis functions, unlike the basis function for DFT and DCT, are well localized in both space and frequency. This expansion onto discrete wavelet basis functions is generally implemented by a digital filterbank using a pair of low-pass and high-pass filters (similar to sub-band coding). Since we almost always employ separable filtering, the computation of 2D-wavelet transform reduces to that of 1D transform. The principles of 1D discrete-wavelet transform and fundamentals of wavelet analysis-synthesis filter design were covered in Section 3.2.4. Here, we review the key points of that discussion that are relevant to image compression.

In 1D binary discrete wavelet analysis, a signal $s(n)$ is split into two equal-size frequency bands, called lower and upper frequency bands, shown in Figure 7.10. If we let the normalized sampling frequency be equal to $f=1$, we need a low-pass filter $H_l(f)$ with the passband $(0, 1/4)$ and a high-pass filter $H_u(f)$ with the passband $(1/4, 1/2)$ for binary decomposition. Since we employ FIR filters, we have to allow an overlap between the passbands of the low-pass and high-pass filters to avoid any frequency gaps. The frequency responses of a realizable pair of low-pass and high-pass filters that exhibit mirror symmetry about $f=1/4$ are depicted in Figure 7.11. Te outputs of these analysis flters are sub-sampled by 2 to obtain the low-pass and high-pass sub-signals, $y_l(n)$ and $y_h(n)$, respectively. Note that, due to sub-sampling, the total number of samples in $y_l(n)$ and $y_h(n)$ is equal to the number of samples in $s(n)$.

After compression and decompression, the processed sub-signals $\hat{y}_l(n)$ and $\hat{y}_h(n)$ are upsampled by zero insertion and then filtered to reconstruct the processed signal $\hat{s}(n)$. The filters $g_l(n)$ and $g_u(n)$ are called synthesis filters. Assuming lossless compression, i.e., $\hat{y}_l(n)=y_l(n)$ and $\hat{y}_h(n)=y_h(n)$, the filters $h_l(n)$, $h_u(n)$, $g_l(n)$, and $g_u(n)$ can be designed such that $\hat{s}(n)=s(n)$.

The fact that the frequency response of the low-pass filter $H_l(f)$ extends into the band $(1/4, 1/2)$ and vice versa causes unavoidable aliasing when each sub-band signal is decimated by 2. In order to achieve alias-free analysis-synthesis filtering (assuming lossless coding), the filters must be designed in such a way that the aliasing introduced by the analysis filter is canceled by the synthesis filter. Hence, the analysis-synthesis filters should satisfy the following properties:

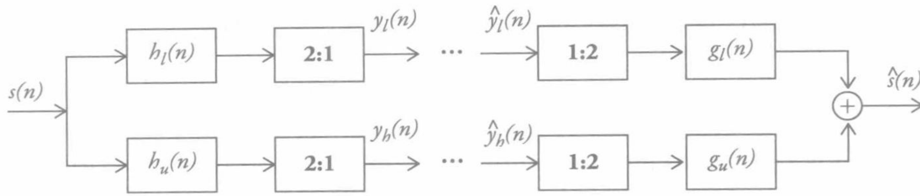



**Figure 7.10** Block diagram of sub-band decomposition and reconstruction filtering.

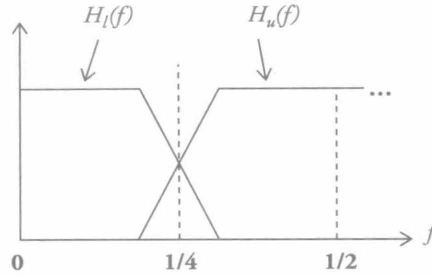



**Figure 7.11** Frequency response of 1D binary decomposition filters.

1. Alias-cancellation: Perfect reconstruction requires that the filters satisfy (see Section 3.2.4)

$$H_l(f)G_l(f) + H_u(f)G_u(f) = 2 \tag{7.7}$$

$$H_l(-f)G_l(f) + H_u(-f)G_u(f) = 0 \tag{7.8}$$

2. Symmetry: An important concern in image compression is to avoid increasing the number of image samples while filtering. Since linear convolution increases the number of samples, filtering is implemented by circular convolution, which yields the same number of samples as the input. We apply symmetric boundary extension in order to avoid introducing unnecessary high-frequency energy due to artificial left-to-right and top-to-bottom intensity discontinuities. However, to preserve the symmetry after filtering so the number of wavelet coefficients to be encoded does not increase, the filter must also be symmetric. Note that odd-length symmetric FIR filters are called zero-phase filters.
3. Orthogonality: Orthogonal filters implement a projection of the input image onto a set of orthogonal wavelet-basis functions. With proper normalization, orthogonal transforms preserve energy and norm, as stated by Parseval's theorem. This property ensures that the energy of the quantization error committed by quantization of the transform coefficients remains unchanged in the pixel domain.

It has been shown that the only filter that satisfies perfect reconstruction, symmetry, and orthogonality is the trivial case of 2-tap Haar filters, $h_l[n]=\{1,1\}$ and $h_u[n]=\{1,-1\}$, which does not have good energy-compaction property. Hence, in order to design FIR filters (compactly supported wavelets) with good energy compaction (a larger number of vanishing moments), we need to give up either orthogonality or symmetry. In modern wavelet image compression, symmetry turns out to be more important than orthogonality, since we can design symmetric, bi-orthogonal (non-orthogonal) filters that are nearly orthogonal.

Bi-orthogonal flters: Te conditions (7.7) and (7.8) can be equivalently stated as

$$H_l(f)G_l(f) + H_l(-f)G_l(-f) = 2$$
$$H_u(f)G_u(f) + H_u(-f)G_u(-f) = 2$$
$$H_l(f)G_u(f) + H_l(-f)G_u(-f) = 0$$
$$H_u(f)G_l(f) + H_u(-f)G_l(-f) = 0$$

which can be expressed in the spatial domain as bi-orthogonality constraints [Vet 92]

$$\langle h_l[k], g_l[2n-k]\rangle = \delta[n]$$
$$\langle h_u[k], g_u[2n-k]\rangle = \delta[n]$$
$$\langle g_u[k], h_l[2n-k]\rangle = 0$$
$$\langle g_l[k], h_u[2n-k]\rangle = 0$$

where $\langle \cdot,\cdot \rangle$ denotes the inner product. In wavelet terms, the filters $\{h_l[n], h_u[n]\}$ are derived from a pair $\{\Phi_1, \Psi_1\}$, while the filters $\{g_l[n], g_u[n]\}$ are derived from another pair $\{\Phi_2, \Psi_2\}$ that are related to $\{\Phi_1, \Psi_1\}$ by the bi-orthogonality constraints. The bi-orthogonality conditions provide us with more flexibility to design odd-length symmetric FIR filters. A complete derivation of wavelet filter design is beyond the scope of this book. The interested reader is referred to [Vai 87] and [Vet 89] for details.

There is a large family of bi-orthogonal wavelets. Among these (9,7) filters are nearly orthogonal and provide good energy compaction. Hence, they were selected to be used in the JPEG2000 standard. The coefficients of (9,7) and (5,3) filters are tabulated in Tables 7.12 and 7.13, respectively. The wavelet filters possess some regularity properties that not all orthogonal QMF filters have, which give them improved coding efficiency over orthogonal QMF filters with the same number of taps. For example, Antonini *et al.* [Ant 92] report that they can achieve a performance close to that of Woods and O'Neil [Woo 86] by using 9/7-tap filters, whereas the latter uses

32-tap Johnston filters.

The 1D decompositions can be extended to two dimensions by using separable filters, i.e., splitting the image $s(n_1, n_2)$ first in the row and then in the column direction, or vice versa. Using a binary decomposition in each direction, we obtain four sub-bands called low (L) $y_L(n_1, n_2)$, horizontal (H) $y_H(n_1, n_2)$, vertical (V) $y_V(n_1, n_2)$, and diagonal (D) $y_D(n_1, n_2)$, corresponding to lower-lower, upper-lower, lower-upper, and upper-upper sub-bands, respectively. The decomposition can be continued by splitting all sub-bands or just the L sub-band repetitively, as shown in Figure 7.12. Other decompositions are also possible.

The wavelet transform coefficients correspond to pixels of respective sub-images. In most cases, the decomposition is carried out in multiple stages. The total number of samples in all sub-images $y_L(n_1, n_2)$, $y_V(n_1, n_2)$, $y_H(n_1, n_2)$, and $y_D(n_1, n_2)$ is the same as the number of samples in the input image $s(n_1, n_2)$ after sub-sampling of the sub-images. Thus, the wavelet decomposition itself does not result in data compression or expansion. Observe that $y_L(n_1, n_2)$ corresponds to a low-resolution version of the image $s(n_1, n_2)$, while $y_D(n_1, n_2)$ contains the high-frequency detail information. Therefore, the wavelet decomposition is also known as a "multi-scale" or "multi-resolution" representation, and can be used in progressive transmission.

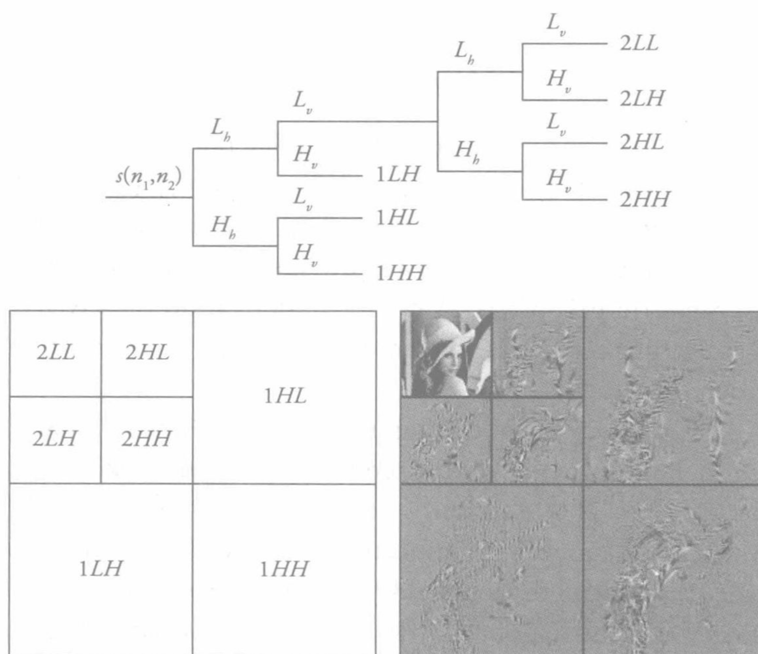



**Figure 7.12** Two-level binary-tree decomposition of an image into frequency bands.

**Sub-Band Coding vs. Wavelet Image Coding**

Sub-band coding quantizes and encodes coefficients in each sub-band independently of other bands. It allocates bit-rates to each sub-band according to a formula based on the variance of the coefficients in that sub-band [Vet 84, Woo 86]. However, the optimal rate allocation changes as the total rate varies and the allocation and encoding must be redone for each rate point. Furthermore, because the rate allocation is based on theoretical quantization models that fit higher bit-rates better, in practice it is difficult to match the desired total target bit-rate. As a result, matching the target bit-rate or a pre-specified filesize involves some trial and error. Finally, the resulting bitstream is not embedded; i.e., bits are not arranged in decreasing order of importance, and truncation of the bitstream may yield unpredictable results. Some early wavelet image encoders employ similar strategies [Bra 94].

Modern wavelet coders leading to the JPEG 2000 standard employ fundamentally different techniques that produce embedded bit-streams [Use 01]. Recent wavelet-compression methods, such as embedded zero tree wavelet transform (EZW) [Sha 93] and set partitioning in hierarchical trees (SPIHT) [Sai 96], do not use explicit bit-allocation formulae and have introduced two main new ideas: i) dependent coding of coefficients across different frequency bands using zero trees and set partitioning in hierarchical trees and ii) embedded bit-stream generation using successive approximation quantization, which both have influenced JPEG 2000.

## 7.3.2 ISO JPEG2000 Standard

Unlike EZW and SPIHT, JPEG2000 does not exploit correlation between sub-bands. JPEG2000, based on the embedded block coding with optimized truncation (EBCOT) [Tau 00], encodes each sub-band independently. It divides an image into tiles (non-overlapping rectangular blocks). A color transform is applied to tiles. Each color component of each tile is input to the block diagram shown in Figure 7.13, where computation of DWT is followed by uniform quantization of the DWT coefficients. The quantized coefficients are then divided into rectangular code blocks, which are entropy coded independently. In Tier-1 (T1) coding, the entropy encoder generates an embedded bitstream for each code block by coding wavelet coefficients bit-plane-by-bit-plane. In Tier-2 coding, a rate-allocation algorithm can be used to aggregate T1 bitstreams into a single interleaved output bitstream in the order of significance of all encoded bits. Each block is described next. Details can be found in [Ima 02, ISO 01].

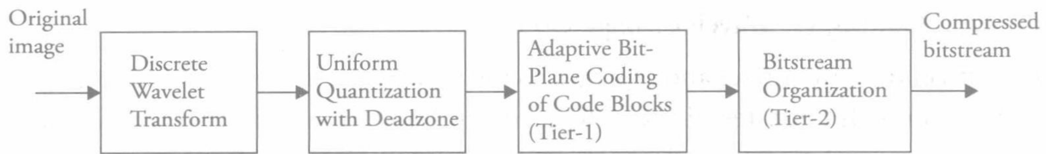



**Figure 7.13** Block diagram of JPEG2000 operation.

## Pre-Processing and Color Transforms

Tiling partitions the input image into rectangular, non-overlapping blocks of equal size (except possibly at the image borders). The tile size is arbitrary and can be as large as the whole image. Each tile is compressed independently with its own set of compression parameters.

Next, unsigned pixel values represented by $B$ bits (in each channel) are level-shifted by subtracting the fixed value $2^{B-1}$ to make pixel values symmetric about 0. Level-shifted pixel values are subject to color transform. JPEG2000 defines two color transforms: irreversible color transform (ICT) and reversible color transform (RCT). ICT is the traditional RGB to YCrCb color transform, used in lossy image compression, given by

$$\begin{aligned} Y &= 0.299R + 0.587G + 0.114B \\ Cr &= 0.713(R - Y) \\ Cb &= 0.564(B - Y) \end{aligned}$$

or equivalently

$$\begin{bmatrix} Y \\ Cr \\ Cb \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.500 & -0.419 & -0.081 \\ -0.169 & -0.331 & 0.500 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

RCT can be used in both lossy or lossless compression, and is given by

$$\begin{aligned} Y &= \left\lfloor \frac{R+2G+B}{4} \right\rfloor \\ U &= R-G \\ V &= B-G \end{aligned}$$

The inverse of RCT is given by

$$G = Y - \left\lfloor \frac{U+V}{4} \right\rfloor$$
$$R = U + G$$
$$B = V + G$$

**Wavelet Filters**

Part 1 of JPEG2000 includes two choices of DWT filterbank. They are the Daubechies (9,7) floating-point filterbank, which provides superior compression performance, and the (5,3) filterbank for use in lifted integer-to-integer (reversible) DWT implementation. The coefficients of the linear-phase FIR filters, $h_0(n)$, $h_1(n)$, $g_0(n)$, $g_1(n)$, are listed in Tables 7.8 and 7.9. Both flterbanks are bi-orthogonal, which means that $h_0(n)$ and $g_1(n)$, are orthogonal and $g_0(n)$ and $h_1(n)$ are orthogonal.

The output wavelet coefficients are floating-point numbers when using floating-point (9,7) DWT filterbanks. The lifting method to compute DWT coefficients provides a way to compute an integer-to-integer DWT. JPEG2000 standard defines conversion of the (5,3) filterbank into an integer-to-integer transform.

**Table 7.8** Daubechies (9,7) Floating-Point Filterbank

| n | Low-pass, $h_0(n)$ | Low-pass, $g_0(n)$ |
|---|---|---|
| 0 | +0.602949018236360 | +1.115087052457000 |
| ±1 | +0.266864118442875 | +0.591271763114250 |
| ±2 | −0.078223266528990 | −0.057543526228500 |
| ±3 | −0.016864118442875 | −0.091271763114250 |
| ±4 | +0.026748757410810 | |

| n | High-pass, $h_1(n)$ | n | High-pass, $g_1(n)$ |
|---|---|---|---|
| −1 | +1.115087052457000 | 1 | +0.602949018236360 |
| −2,0 | −0.591271763114250 | 0,2 | −0.266864118442875 |
| −3,1 | −0.057543526228500 | −1,3 | −0.078223266528990 |
| −4,2 | +0.091271763114250 | −2,4 | +0.016864118442875 |
| | | −3,5 | +0.026748757410810 |

Table 7.9 (5,3) Filterbank

| n | $h_0(n)$ | $g_0(n)$ | n | $h_1(n)$ | n | $g_1(n)$ |
|---|---|---|---|---|---|---|
| 0 | +3/4 | +1 | −1 | +1 | 1 | +3/4 |
| ±1 | +1/4 | +1/2 | −2,0 | −1/2 | 0,2 | −1/4 |
| ±2 | −1/8 | | | | −1,3 | −1/8 |

### Filter Normalization

DWT filters are often normalized in terms of DC gain

$$G_{DC} = \left|\sum_n h(n)\right|$$

for low-pass filters and the Nyquist gain

$$G_{Nyq} = \left|\sum_n (-1)^n h(n)\right|$$

for high-pass filters. The (9,7) and (5,3) analysis filterbanks have been normalized such that $G_{DC}=1$ and $G_{Nyq}=2$. This is referred to as (1,2) normalization, which is adopted in Part 1 of JPEG 2000. Other forms that appear in the literature are $\left(\sqrt{2},\sqrt{2}\right)$ and (1,1) normalizations. Once normalization of the analysis filterbank is specified, normalization of synthesis filterbank is automatically determined by reversing the order and multiplying by a scalar factor $c$.

### Boundary Extension

In order to cope with image border effects, each line of the image is extended by the symmetric and periodic boundary extension scheme when using odd-tap filters as shown in Figure 7.14.

### Quantization at the Encoder

Similar to the original JPEG standard, JPEG2000 employs uniform quantization of the wavelet coefficients, with one step-size for each sub-band. An important difference is in the inclusion of a central deadzone. It has been shown that the R-D optimal quantizer for a signal with Laplacian probability density (such as DCT or wavelet coefficients) is a uniform quantizer with a central deadzone. The size of the optimal deadzone as a fraction of the step-size increases as the variance of the Laplacian distribution decreases. In Part 1, the size of the deadzone is taken as twice the step-size due to its optimal embedded structure. Tis is shown in Figure 7.15.

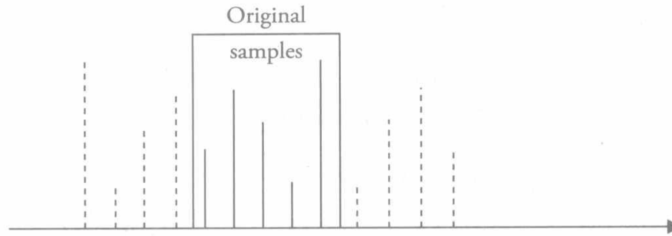



**Figure 7.14** Symmetric and periodic boundary extension.




**Figure 7.15** Uniform quantizer with a deadzone.

Embedded quantization means that an $M_b$ bit quantizer index resulting from step-size $\Delta_b$ is transmitted progressively starting with the most-significant bit (MSB) and proceeding toward the least-significant bit (LSB). The resulting index after decoding only $N_b$ bits corresponds to a quantizer with a step-size $\Delta_b 2^{M_b - N_b}$.

The quantization at the encoder is performed according to

$$q_b(k_1,k_2) = sign(y_b(k_1,k_2)) \left\lfloor \frac{|y_b(k_1,k_2)|}{\Delta_b} \right\rfloor$$

where the step-size $\Delta_b$ is represented by an 11-bit mantissa $\mu_b$ and 5-bit exponent $\varepsilon_b$ as follows:

$$\Delta_b = 2^{R_b - \varepsilon_b}\left(1 + \frac{\mu_b}{2^{11}}\right)$$

where $R_b$ is the number of bits used to represent the nominal dynamic range of sub-band $b$.

JPEG2000 allows two modes to signal the value of $\Delta_b$ to the decoder:

1. Expounded quantization: One $(\varepsilon_b, \mu_b)$ value for every sub-band is explicitly transmitted.
2. Derived quantization: A single $(\varepsilon_0, \mu_0)$ value is sent for the LL sub-band. Values for other sub-bands $(\varepsilon_b, \mu_b)$ are derived by scaling $\Delta_0$ value

   $$(\varepsilon_b, \mu_b) = (\varepsilon_0 - N_L + n_b, \mu_0)$$

   where $N_L$ is the total number of decomposition levels and $n_b$ is the decomposition level for sub-band $b$.

**Inverse Quantization at the Decoder**

The inverse quantization is performed by biased sample reconstruction (instead of the usual mid-point reconstruction) defined by

$$Rq_b(k_1,k_2) = \begin{cases} (q_b(k_1,k_2)+\gamma)\Delta_b & \text{if } q_b(k_1,k_2)>0 \\ (q_b(k_1,k_2)-\gamma)\Delta_b & \text{if } q_b(k_1,k_2)<0 \\ 0 & \text{otherwise} \end{cases}$$

where $0 \leq \gamma < 1$ is a bias parameter. $\gamma = 0.5$ corresponds to mid-point reconstruction. A value of $\gamma < 0.5$ creates a bias towards zero, which results in improved reconstruction PSNR when probability distribution of wavelet coefficients falls off rapidly away from zero. A popular choice is $\gamma = 0.375$.

**Entropy Coding**

JPEG2000 constructs an embedded bitstream by bit-plane encoding of quantizer indices. Bit-plane coding of wavelet coefcients, which is illustrated in Figure 7.16, has also been used by other embedded wavelet coders such as EZW and SPIHT. JPEG 2000 uses a block-coding paradigm in the wavelet domain, where each subband is partitioned into small rectangular blocks, called code blocks, that are also coded independently. While this results in a small loss of compression efficiency, it has several other benefits such as improved error resilience, flexible arrangement of progression orders, localized random access, and improved cropping.

During progressive encoding of bit-planes, a quantized wavelet coefficient is called insignificant if the quantizer index bit is still zero. The coefficient becomes significant when the first non-zero bit is encountered, then the sign of the coefficient is also encoded. Once a coefficient becomes significant, all subsequent bits are referred to as refinement bits. All bits are encoded using context-based adaptive binary arithmetic coding to form the Tier-1 bitstream.

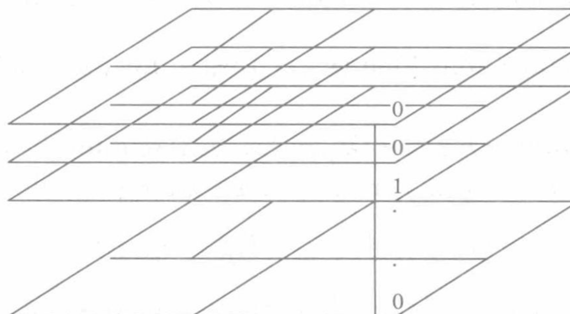



**Figure 7.16** Progressive encoding of bit-planes.

**Region-of-Interest Coding**

Region-of-interest (ROI) coding has two functionalities: i) to compress the ROI with higher fidelity when the full bitstream is decoded, and ii) to transmit the ROI before the background region in case the bitstream is partially decoded. Both functionalities are achieved by the Maxshift method. The basic principle of the Maxshift method is to scale up (shift up) the wavelet coefficients in the ROI. In particular, the bits associated with the ROI are shifted up such that all the bits of wavelet coefficients in the ROI are in higher bit-planes than the most significant bit of all other pixels within that tile. During the embedded coding process, coded bits for higher bit-planes are placed prior to the background bits in the final bitstream. No shape information for the ROI needs to be included in the bitstream.

**Error Resilience**

Error resilience tools include compressed data partitioning, resynchronization, and error detection. Error-resilient bitstream syntax and tools are provided both at the entropy coding and packetization levels. The independent code-block coding approach employed in JPEG2000 leads to improved error resilience. Finally, JPEG2000 specifies an optional file format called JP2.

## References

[Ahm 74] Ahmed, N., T, Natarajan, and K. R. Rao, "Discrete cosine transforms," IEEE Trans. on Comp., vol. 23, pp. 90–93, 1974.

[Ant 92] Antonini, M., M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," IEEE Trans. Image Proc., vol. 1, pp. 205–220, Apr. 1992.

[Ber 71] Berger, T., *Rate Distortion Theory*, Englewood Cliffs, NJ: Prentice Hall, 1971.

[Bra 94] Bradley, J., and C. Brislawn, "The wavelet scalar quantization compression standard for digital fingerprint images," Proc. IEEE Int. Symp. Circ. and Systems (ISCAS), pp. 205–208, London, UK, June 1994.

[Bur 83] Burt, P., and E. Adelson, "The Laplacian pyramid as a compact image code," IEEE Trans. Comm., vol. 31, pp. 482–540, 1983.

[Che 84] Chen, W.-H., and W. K. Pratt, "Scene adaptive coder," IEEE Trans. Comm., vol. 32, pp. 225–232, Mar. 1984.

[Ger 92] Gersho, A., and R.M. Gray, *Vector Quantization and Signal Compression*, Boston, MA: Kluwer, 1992.

[Gon 07] Gonzalez, Rafael C., and Richard E. Woods, *Digital Image Processing, Third Edition*, Upper Saddle River, NJ: Prentice Hall, 2007.

[Gra 98] Gray, R. M., and D. L. Neuhoff, "Quantization," IEEE Trans. on Information Theory, vol. 44, no. 6, pp. 2325–2383, Oct. 1998.

[How 98] Howard, P., F. Kossentini, B. Martins, S. Forchhammer, and W. Rucklidge, "The emerging JBIG2 standard," IEEE Trans. on Circuit and Syst. for Video Tech., vol. 8, no. 5, pp. 838–848, Sept. 1998.

[Ima 02] Image Communication, Special Issue on JPEG2000, vol. 17, no. 1, Jan. 2002.

[ISO 01] ISO/IEC 15444–1, ITU Rec. T.800, JPEG 2000 Image Coding System, 2001.

[Llo 82] Lloyd, S.P., "Least squares quantization in PCM," IEEE Trans. Info. Theory, vol. 28, pp. 129–137, Mar. 1982.

[Mar 94] Martucci, S., "Symmetric convolution and the discrete sine and cosine transforms," IEEE Trans. Signal Proc., vol. 42, pp. 1038–1051, May 1994.

[Max 60] Max, J., "Quantizing for minimum distortion," IRE Trans. Info. Theory, vol. 6, pp. 7–12, Mar. 1960.

[Net 88] Netravali, A.N., and B. G. Haskell, *Digital Pictures: Representation and Compression,* New York, NY: Plenum Press, 1988.

[Pen 88] Pennebaker, W. B., J. L. Mitchell, G. G. Langdon, Jr., R. B. Arps, "An overview of the basic principles of the Q-Coder adaptive binary arithmetic coder," IBM J. of Research and Dev., vol. 32, no. 6, p. 717, 1988.

[Pen 93] Pennebaker, W. B., and J. L. Mitchell, *JPEG: Still Image Data Compression Standard*, New York, NY: Van Nostrand Reinhold, 1993.

[Rab 91] Rabbani, M., and P. Jones, *Digital Image Compression Techniques*, Bellingham, WA: SPIE Press, 1991.

[Rao 90] Rao, K. R., and P. Yip, *Discrete Cosine Transform*, Boston, MA: Academic Press, 1990.

[Sai 96] Said, A., and W. Pearlman, "A new fast and efficient image codec based on set partitioning," IEEE Trans. on Circ. and Syst. for Video Tech., vol. 6, pp. 243–250, June 1996.

[Say 02] Sayood, K., *Lossless Compression Handbook,* Boston, MA: Academic Press, 2002.

[Sha 48] Shannon, C. E., "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, pp. 379–423 (July) and pp. 623–656 (October), 1948.

[Sha 93] Shapiro, J., "Embedded image coding using zero-tree of wavelet coefficients," IEEE Trans. on Signal Processing, vol. 41, pp. 3445–3462, Dec. 1993.

[Tau 00] Taubman, D., "High performance scalable image compression with EBCOT," IEEE Trans. on Image Proc., vol. 9, pp. 1158–1170, July 2000.

[Use 01] Usevitch, B. E., "A tutorial on modern lossy wavelet image compression: Foundations of JPEG 2000," IEEE Signal Proc. Magazine, pp. 22–35, Sept. 2001.

[Vai 87] Vaidyanathan, P. P., "Quadrature mirror filterbanks, M-band extensions, and perfect reconstruction technique," IEEE Acoust. Speech Sign. Proc. Magazine, vol. 4, pp. 4–20, 1987.

[Vet 84] Vetterli, M., "Multidimensional subband coding: Some theory and algorithms," Signal Processing, vol. 6, pp. 97–112, Feb. 1984.

[Vet 89] Vetterli, M., and D. LeGall, "Perfect reconstruction FIR filterbanks: Some properties and factorizations," IEEE Trans. Acoust. Speech Sign. Proc,, vol. 37, pp. 1057–1071, 1989.

[Vet 92] Vetterli, M., and C. Herley, "Wavelets and filterbanks: theory and design," IEEE Trans. on Signal Processing, vol. 40, pp. 2207–2231, Sept. 1992.

[Wal 91] Wallace, G.K., "The JPEG still picture compression standard," Commun. of the ACM, vol. 34, pp. 30–44, 1991.

[Wei 00] Weinberger, M., G. Seroussi, and G. Sapiro, "The LOCO-I lossless image compression algorithm: Principles and standardization into JPEG-LS," IEEE Trans. Image Processing, vol. 9, no. 8, pp. 1309–1324, Aug. 2000.

[Wit 87] Witten, I., R. Neal, and J. Cleary, "Arithmetic coding for data compression," Comm. of the ACM, vol. 30, no. 6, pp. 520–540, 1987.

[Woo 69] Wood, R. C., "On optimum quantization," IEEE Trans. Info. Theory, vol. 15, no. 2, pp. 248–252, 1969.

[Woo 86] Woods, J.W., and S. O'Neil, "Subband coding of images," IEEE Trans. on Acoust. Speech and Signal Proc., vol. 34, pp. 1278–1288, Oct. 1986.

[Ziv 77] Ziv, J., and A. Lempel, "A universal algorithm for sequential data compression," IEEE Trans. Info. Theory, vol. IT-23, pp. 337–343, 1977.

[Ziv 94] Ziv, J., "On universal data compression - An intuitive overview," J. Vis. Comm. Image Rep., vol. 5, no. 4, pp. 317–321, Dec. 1994.

## Exercises

**7.1** Suppose we have a DMS with the alphabet $\boldsymbol{A}$ and the symbol probabilities $p(a_i)$, $a_i \in \boldsymbol{A}$ specified by the following table:

| $a_0$ | $a_1$ | $a_2$ | $a_3$ | $a_4$ | $a_5$ |
|---|---|---|---|---|---|
| 0.3 | 0.2 | 0.2 | 0.1 | 0.1 | 0.1 |

**a.** Find the entropy of this source.

**b.** Design a Huffman code for this source.

c. Find the average codeword length.
d. How good is this code?
e. Can Huffman coding result in data expansion? Explain why or why not.

**7.2** Suppose we have a binary source with symbol probabilities P(0)=0.75 and P(1)=0.25.
a. Design an arithmetic code for the following sequence of symbols:

$$0000110010$$

b. What is the entropy of the source?
c. What is the average codeword length for this instance of the source? Explain.

**7.3** Explain why the ITU Group 3 and Group 4 codes may result in data expansion with half-tone images. How does the JBIG standard address this problem?

**7.4** What is the primary motivation of using the Gray codes in bit-plane encoding?

**7.5** Suppose a random variable $X$ that is uniformly distributed between 0 and 10 is uniformly quantized with $N$ levels.
a. Find all decision and reconstruction levels.
b. Calculate the mean-square quantization error.

**7.6** Suppose we have three scalar random variables, $X_1$, $X_2$, and $X_3$, with variances 2.5, 7, and 10. Propose a method to allocate a total of 20 bits to quantize each random variable independently to minimize the mean of the sum of the square-quantization errors.

**7.7** Let $X_n$, $n=\ldots,-1,0,1,\ldots$ denote a sequence of zero-mean scalar random variables with

$$E\left\{X_n^2\right\}=\sigma_X^2 \text{ and } E\left\{X_n X_{n-1}\right\}=\alpha$$

We define the prediction-error sequence

$$\epsilon_n = X_n - \rho X_{n-1}$$

where $\rho$ is such that

$$E\left\{(X_n-\rho X_{n-1})X_n\right\}=0 \text{ for all } n$$

a. Find the value of $\rho$.

**b.** Let $D_1$ and $D_2$ denote the distortion incurred by uniform quantization of $X_n$ and $\epsilon_n$, respectively, using $r$ bits. Show that $D_1$ is always greater than or equal to $D_2$.

**7.8** Compute the Karhunen–Loeve transform, DFT, and DCT of the block shown below.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 16 | 11 | 10 | 16 | 24 | 40 | 51 | 61 |
| 12 | 12 | 14 | 19 | 26 | 58 | 60 | 55 |
| 14 | 13 | 16 | 24 | 40 | 57 | 69 | 56 |
| 14 | 17 | 22 | 29 | 51 | 87 | 80 | 62 |
| 18 | 22 | 37 | 56 | 68 | 109 | 103 | 77 |
| 24 | 35 | 55 | 64 | 81 | 104 | 113 | 92 |
| 49 | 64 | 78 | 87 | 103 | 121 | 120 | 101 |
| 72 | 92 | 95 | 98 | 112 | 100 | 103 | 99 |

Compare these transforms on the basis of energy compaction (decorrelation) and similarity of the basis functions.

**7.9** Explain why zigzag scanning is used in DCT compression.

**7.10** How do you achieve bit-rate (bits/pixel) vs. quality (PSNR) tradeoff in JPEG image coding?

**7.11** How does the JPEG algorithm take advantage of spectral (color) and perceptual (psychovisual) redundancy?

**7.12** Elaborate on the relationship between the subband coding and the DCT coding. In particular, consider a 64×64 image. Suppose we partition the image into 8×8 blocks and compute their DCT.

**a.** Show that we can construct 64 subband images, each of which is 8×8, by an ordering of the DCT coefficients.

**b.** Discuss the relationship between DPCM encoding of the DC coefficients in JPEG and DPCM encoding of the lowest subband in this case.

**7.13** Show that a perfect reconstruction filterbank cannot satisfy the following requirements simultaneously: i) it is FIR symmetric (zero-phase) with filter length greater than 2 and ii) analysis and synthesis filters are orthogonal.

## Internet Resources

J. Abel, The Data Compression Resource on the Internet
http://www.data-compression.info

M. Dipperstein, Lempel-Ziv-Welch Encoding
http://michael.dipperstein.com/lzw/

Markus Kuhn, JBIG-KIT (JBIG1)
http://www.cl.cam.ac.uk/~mgk25/jbigkit/

JBIG2 Encoder and Decoder
http://www.ghostscript.com/jbig2dec.html
https://github.com/agl/jbig2enc

CharLS, a JPEG-LS library
http://charls.codeplex.com/

OpenJPEG Homepage—JPEG2000 codec
http://www.openjpeg.org/

Image and Video Compression Learning Tool VCDemo, TU Delft, 2011
http://insy.ewi.tudelft.nl/content/image-and-video-compression-learning-tool-vcdemo

CHAPTER 8

# Video Compression

*Video compression is a critical technology that has emerged over the past four decades and has enabled complete transition from analog to digital video in all entertainment and communication industries, including digital TV, digital cinema, and visual communications. It has revolutionalized how we consume and communicate visual media, making the Internet the premiere media and visual information-exchange environment.*

The high bit-rate requirements for uncompressed standard and high-definition video formats make the need for video compression evident. Different industries and applications may have different video resolution and quality requirements, ranging from lossless compression for studio editing and archiving to lossy compression at various target bit-rate and quality points depending on the available display size and resolution as well as transmission bit-rate. In order to respond to these varying requirements, a number of compression algorithms and standards have been developed.

The simplest approach to video compression would be to employ a still-frame compression technique, such as JPEG or JPEG 2000, on a frame-by-frame basis to provide random access to any frame. However, the compression ratio that can be achieved by this approach is limited because inter-frame (temporal) redundancies are ignored. Inter-frame-compression methods, which take advantage of temporal redundancy by exploiting similarity between neighboring frames, e.g., through

motion-compensated (MC) coding, provide superior compression efficiency. This chapter first presents different video-compression approaches, including a brief discussion on how to best exploit the temporal redundancy, and then introduces various international standards that employ the MC-transform-coding strategy.

## 8.1 Video-Compression Approaches

A variety of approaches, ranging from MC transform coding and 3D-transform coding, including discrete cosine transform (DCT) and wavelet transform, to vector quantization, fractal coding, and 2D and 3D model-based coding, have been investigated for video compression in the literature. A comprehensive review of model-based coding techniques can be found in [Aiz 95]. While some of these techniques have been successful for specific types of video at very low bit-rates, we only discuss the mainstream approaches of intra-only coding, 3D-transform coding, and block-motion compensated coding. Indeed, modern video-coding applications and standards employ only intra-coding and MC transform coding.

### 8.1.1 Intra-Frame Compression, Motion JPEG 2000, and Digital Cinema

Intra-frame compression means each picture is encoded independently using the image-compression techniques discussed in Chapter 7; i.e., compressed pictures do not depend on any data from the preceeding or succeeding pictures. Intra-only compression has some advantages such as:

- It enables random access to each frame, and
- It has low complexity (no multiple frame stores and motion estimation).

However, the compression efficiency of intra-only methods is limited because inter-frame (temporal) redundancies cannot be exploited. Nevertheless, intra-only coding has found applications in consumer/professional media and digital cinema capture, studio editing, and archiving.

The consumer digital video format DV and its variants DVCPRO and DVCAM used in camcorders as well as DVCPRO-50 and DVCPRO HD used in professional television production employ intra-only video coding at 25 Mbits/s to 50 Mbits/s data streams. They all use DCT-based JPEG-like intra-compression. The DV algorithm yields higher quality than the JPEG scheme at the nominal 5:1 compression ratio by allowing for optimization of quantizing tables within a frame. It also uses

adaptive inter-field compression; i.e., it compresses the two fields of an interlaced frame together if little or no motion is detected to allow for higher overall quality.

Motion JPEG 2000 (MJ2K) refers to using the JPEG 2000 still image-compression method for a motion picture or video sequence in a frame-by-frame manner. In addition to achieving better compression efficiency than the DCT-based JPEG, JPEG 2000 provides the ability to support lossless, near lossless, and lossy encoding in a single embedded bit-stream, which makes MJ2K attractive for the motion-picture industry to use in the entire workflow, including digital production, post-production, projection, and archiving. The compliance point 3 of MJ2K covers production and projection formats, with image sizes up to 4096 × 3112, 4:4:4 color with up to 16 bits/color, and up to five transform layers. The compression efficiency of MJ2K was significantly favorable compared to using an intra-only AVC/H.264 video codec for high- and ultra-high-definition video [Mar 03].

## 8.1.2 3D-Transform Coding

3D-transform coding refers to straightforward extension of both 2D-DCT and wavelet-based coding approaches to video encoding using spatio-temporal (3D) transforms.

### 3D-DCT Coding

In 3D-DCT coding, video is divided into $M \times N \times J$ blocks, as depicted in Figure 8.1 where $M$, $N$, and $J$ denote the horizontal, vertical, and temporal dimensions of the block, respectively. The transform coefficients are quantized and encoded similar to 2D-transform coding of still-frame images. Since the DCT coefficients are related

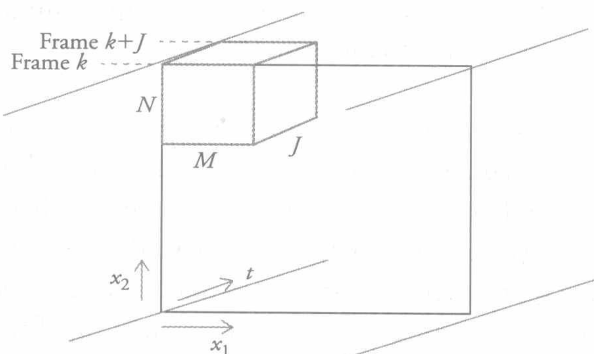



**Figure 8.1** 3D-transform block in 3D-transform coding.

to the frequency content of blocks, the energy for most blocks will be packed in the low-spatial and temporal-frequency zone, and for temporally stationary blocks, the temporal DCT coefficients will be close to zero and will be truncated. The 3D-DCT coding does not require a separate motion-estimation step. However, it requires $J$ frame stores both in the encoder and decoder. Therefore, $J$ is typically chosen as 2 or 4 to allow for inexpensive hardware implementations. Note that random access to video is possible once for every $J$th frame.

The hybrid-transform/DPCM coding strategy implements differential pulse-code modulation (DPCM) of 2D-transform coefficients in the temporal direction to overcome the multiple frame-store requirement of 3D-transform coding [Roe 77]. It employs a 2D-DCT on each spatial block within a given frame. Then, a bank of DPCM coders, each tuned to the statistics of a specific DCT coefficient, is applied to the transform coefficients in the temporal direction. That is, differences in the respective DCT coefficients in the temporal direction are quantized and encoded. Results comparable to that of 3D-DCT coding can be obtained with proper adaptation of the DPCM quantizers to the temporal statistics of the 2D-DCT coefficients [Roe 77]. Note that neither 3D-DCT coding nor hybrid DCT/DPCM coding has been widely used in practical applications.

### 3D-Wavelet/Sub-Band Coding

The development of 3D-wavelet/sub-band coding has been encouraged by the fact that: i) it often does not cause blocking artifacts, which is a common problem with 3D-DCT and MC-DCT coding especially at low bit-rates, ii) unlike MC-compression methods, it does not require a motion-estimation stage, and iii) it is inherently scalable, both spatially and temporally. Scalability, which refers to accessing digital video at various spatial and temporal resolutions without having to decompress the entire bit-stream, has become an important property due to the growing need for storage and transmission of digital video in various definition standards.

In 3D-wavelet/sub-band coding, video is decomposed into a number of properly sub-sampled component signals, ranging from a low spatial and temporal-resolution component to various higher-frequency detail components. These component video signals are encoded using algorithms adapted to the statistical and pyschovisual properties of the respective spatio-temporal frequency bands. Compression is achieved by quantization of the various components and entropy coding of the quantized values. Higher-resolution video, in both spatial and temporal coordinates, can be recovered by combining the decompressed low-resolution and detail components.

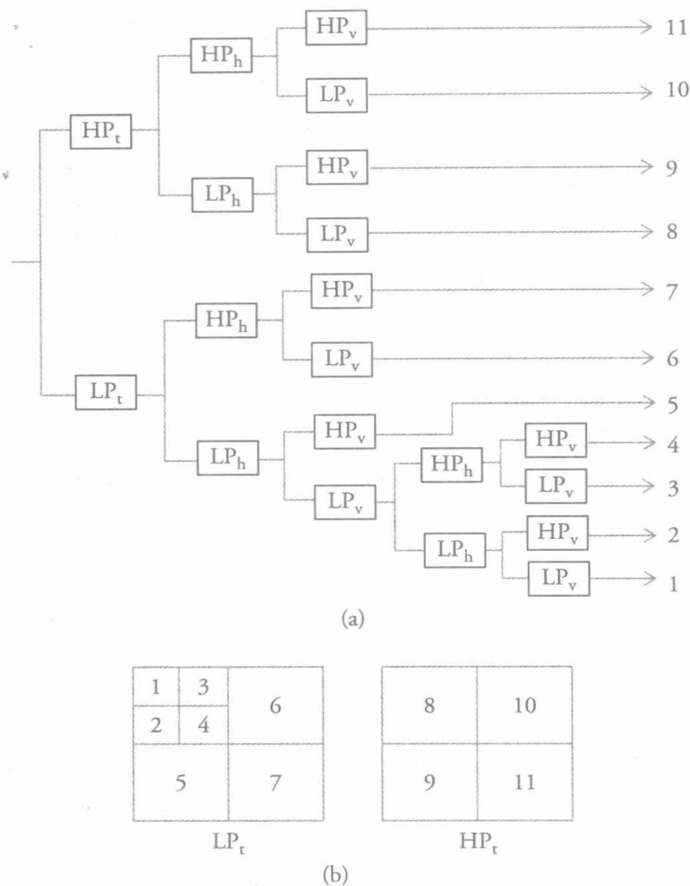



**Figure 8.2** 3D-wavelet decomposition with 11 bands: (a) system-block diagram and (b) spatial and temporal bands, where $LP_t$ and $HP_t$ denote LP and HP temporal bands, respectively.

Most 3D-wavelet/sub-band decompositions use 2- or 4-frame temporal blocks at a time due to practical implementation considerations. Typically, the temporal decomposition is based on a simple 2-tap Haar filterbank [Luo 94], which gives the average and difference of two frames for the low-pass (LP) and high-pass (HP) temporal bands, respectively. In the second stage, both LP and HP temporal sub-bands are decomposed into LP and HP horizontal sub-bands, respectively. In the next stage, each of these bands are decomposed into LP and HP vertical sub-bands. Subsequently, the LP temporal-LP horizontal-LP vertical band is further decomposed into four spatial sub-bands to yield an 11-band decomposition, as depicted in Figure 8.2(a). Spatial decomposition of the LP and HP temporal bands are depicted in Figure 8.2(b). Longer wavelet filters can be applied for spatial (horizontal and vertical) decompositions, since these filters can be operated in parallel and do not affect

frame-store requirements. Various approaches to compressing individual component signals are discussed in [Vet 92, Bos 92, Luo 94, Cho 99].

### 8.1.3 Motion-Compensated Transform Coding

The main idea in MC video compression is to encode the "new" information that is not present in the previously encoded frames. The earliest such approach was the so-called conditional replenishment (CR) technique [Has 72], which segments each frame into "changed" and "unchanged" regions with respect to the previous frame, and encodes only addresses and intensities of pixels in the changed regions. Since the amount of changed information varies from frame to frame, the information to be transmitted needs to be buffered, and quantization should be regulated according to buffer fullness. Note that conditional replenishment is a motion-detection based algorithm rather than an MC algorithm, since it does not require explicit estimation of the motion vectors. It is the basis of the "skip" mode in the modern video-compression standards.

MC techniques characterize the temporal correlation in a video by using motion vectors rather than through the respective transform coefficients. The earliest MC compression scheme was the MC-DPCM, which extends the CR method to encode the displaced frame difference for pixels in the changed region with respect to the previous frame. It yields better compression than CR provided that displacement vectors can be accurately estimated. MC-DPCM uses pixel (pel) recursive algorithms for motion estimation.

The most popular MC-coding approach is MC-transform coding, where the temporal-prediction error after MC (displaced-block difference) is 2D-DCT encoded. The temporal prediction aims at minimizing the temporal redundancy, while the DCT encoding makes use of the spatial redundancy in the prediction error. MC-transform coding algorithms feature several modes to incorporate both progressive and interlaced inputs. These include intra-field, intra-frame, and inter-field and inter-frame prediction with or without motion compensation. In the inter-field and inter-frame modes the prediction is based on the previous field or frame, respectively. The basic MC-transform coding scheme employs block-based motion estimation and compensation. It has been argued that block-motion models are not realistic for most image sequences, since moving objects hardly ever manifest themselves as rectangular image blocks. Recently, variable block-size motion-compensation with more than a single motion vector per macroblock has been shown effective without increasing motion vector overhead. Almost all international standards for video compression employ the MC coding strategy.

# 8.2 Early Video-Compression Standards

## 8.2.1 ISO and ITU Standards

There are two major international organizations that develop video-compression standards: International Telecommunications Union (ITU-T) (formerly CCITT) and International Standards Organization (ISO) in collaboration with the International Electro-technical Committee (IEC). The ITU-T Video Coding Experts Group (VCEG) used to deal with bi-directional real-time video-communications standards, and ISO/IEC Moving Picture Experts Group (MPEG) used to deal with video compression from the viewpoint of information technology. Considering the significant overlap and similarity between these standards, the two organizations recently joined forces to develop joint standards. Table 8.1 presents international video-compression standards in chronological order of their development. MC-transform coding is the basis of all video-compression standards, which are summarized in Table 8.1.

Historically, the first video-compression standard based on MC-transform coding was the ITU-T Recommendation (Rec.) H.261 in order to enable videophone and videoconferencing services over the integrated services digital network (ISDN) at $p \times 64$ kbps, $p = 1, \ldots, 30$ [Che 93]. Rec. H.261 emerged as a result of studies performed within the European COST (CoOperation in the field of Scientific and Technical research) Action 211 during 1983–1990. In 1985, COST 211bis developed a codec operating at bit rates of $n \times 384$ kbps, $n = 1, \ldots, 5$, which was adopted by ITU-T as Rec. H.120 in 1987. Later, it became clear that a single standard can cover all ISDN rates, $p \times 64$ kbps, $p = 1, \ldots, 30$. In addition to forming the basis for later video-compression standards such as MPEG-1 and MPEG-2, Rec. H.261 offers some important features: i) Unlike JPEG, it does not define specific encoders to produce valid bit-streams; instead, flexibility is allowed in designing conformant encoders. ii) It introduces the MC block DCT architecture, which is amenable to

**Table 8.1** International Standards for Video Compression

| Standard | Approved By | First Edition | Short Description |
|---|---|---|---|
| H.261 | ITU-T | 1988 | Obsolete |
| MPEG-1 | ISO/IEC | 1993 | Obsolete |
| MPEG-2/H.262 | ISO/IEC, ITU-T | 1996 | Digital broadcast |
| H.263 | ITU-T | 1996 | Obsolete |
| MPEG4-AVC/H.264 | ISO/IEC, ITU-T | 2003 | All purpose |
| MPEG HEVC/H.265 | ISO/IEC, ITU-T | 2013 | All purpose |

low-cost hardware implementations. iii) It is limited by a maximum coding delay of 150 msec., since it is intended for bi-directional video communication. It has been observed that delays exceeding 150 msec. do not give the viewer impression of direct visual feedback. Although Rec. H.261 is obsolete by now, it influenced future video-coding standards, which we discuss in detail below.

### 8.2.2 MPEG-1 Standard

MPEG-1 is an ISO/IEC standard that was developed for storage of digital video and its associated audio at about 1.5 Mbps on various digital-storage media such as CD-ROM, digital audio tape (DAT), and optical drives. The main parts of the MPEG-1 standard are Systems (ISO/IEC 11172-1), Video (ISO/IEC 11172-2), and Audio (ISO/IEC 11172-3). Here, we concentrate on MPEG-1 video. MPEG became active in 1988. The definition of the video algorithm (committee draft) was completed in 1990. MPEG-1 was approved as an international standard in early 1993. MPEG-1 is a generic standard in that it standardizes the *syntax* for representation of an encoded bit-stream and a method of decoding, including motion-compensated prediction (MCP), discrete-cosine transformation (DCT), quantization, and variable-length coding (VLC). MPEG-1 does not standardize a motion-estimation algorithm or a method for selecting the compression mode. The parameters defining the coded bit-stream and decoders are contained in the bit-stream itself. This allows it to be used with pictures of various sizes and aspect ratios at a range of bit-rates. The quality of MPEG-1 compressed/decompressed video at about 1.2 Mbps (video rate) was found to be similar (or superior) to that of VHS-recorded analog video [Gal 91, Chi 95].

The MPEG-1 video compression is similar to that of H.261 with the following new features: i) MPEG-1 offers random-access points to stored video sequences by introducing I-frames, which consist of only intra-coded macroblocks. In addition, MPEG-1 supports trick modes (fast-forward and fast-reverse functions) for digitally stored video. ii) Two other frame types have also been introduced: P-pictures that are coded in reference to a previous I- or P-picture and B-pictures that are coded by bi-directional MC using a previous and a future reference picture (both possibly multiple frames away from the current picture). In contrast, H.261 does not support bi-directional MC and allows MC only over the immediately previous frame. iii) MPEG-1 supports half-pixel MC and no loop filter. iv) MPEG-1 features visually weighted quantization of DCT coefficients. v) MPEG-1 has a flexible slice structure. vi) MPEG-1 allows for separate VLC tables for intra- and inter- (P or B) coded macroblocks. vii) There are no picture size or bit-rate restrictions except for the constrained parameters. MPEG-1 also supports flexible frame

rates. viii) MPEG-1 offers a reasonable coding/decoding delay of about 1 sec to provide unidirectional interactive access to video. The coding delay in H.261 was strictly limited to 150 msec to maintain bi-directional interactivity. We elaborate on some of these features in the following.

**Input-Video Format and Data Structure**

MPEG-1 allows only progressive (non-interlaced) video as input. Therefore, 525/30 and 625/25 interlaced video must be converted into standard input format (SIF), which is either $352 \times 240$ at 30 frames or $352 \times 288$ at 25 frames (both progressive) for 525/30 and 625/25 analog video, respectively. The smaller spatial dimensions are required in order to reach the target bit-rate of 1.5 Mbps with acceptable video quality. The (Y,Cr,Cb) color space was adopted, as in ITU-R (CCIR) Recommendation 601. In the MPEG-1 SIF, the chroma components are sub-sampled by 2 in both the horizontal and vertical directions (4:2:0 format).

The MPEG-1 bit-stream follows a hierarchical data structure, consisting of the following six layers, which enables the decoder to interpret the data unambiguously:

1. *Sequences* are formed by several groups of pictures.
2. *Groups of pictures* (GOP) are made up of pictures. A GOP of size $N$ contains $N$ pictures.
3. *Pictures* consist of slices. There are four picture types indicating the respective modes of compression: I-pictures, P-pictures, B-pictures, and D-pictures. I-pictures consist of only intra-frame DCT-encoded macroblocks. They serve as random-access points to the video. There are two types of inter-frame-encoded pictures: P- and B-pictures. These pictures contain MC predictive-encoded macroblocks. Only forward prediction is allowed in the P-pictures, which are always encoded relative to the preceding I- or P-pictures. The prediction of B-pictures can be forward, backward, or bi-directional relative to other I- or P-pictures. D-pictures contain only the DC component of each block and facilitate video browsing at very low bit-rates. The number of I-, P-, and B-pictures in a GOP are application-dependent, e.g., dependent on access-time and bit-rate requirements. The number of B-pictures between successive reference (anchor) pictures is denoted by $M-1$, where $M$ is the distance between the anchor pictures (known as prediction distance).
4. *Slices* are made up of macroblocks. They are introduced for error recovery.
5. A *macroblock* (MB) consists of four $8 \times 8$ luminance blocks and the spatially associated chroma blocks similar to JPEG. Some compression parameters can

**Table 8.2** Macroblock Types in MPEG-1

| I-pictures | P-pictures | B-pictures |
|---|---|---|
| Intra | Intra | Intra |
| Intra-A | Intra-A | Intra-A |
| | Inter-D | Inter-F |
| | Inter-DA | Inter-FD |
| | Inter-F | Inter-FDA |
| | Inter-FD | Inter-B |
| | Inter-FDA | Inter-BD |
| | Skip | Inter-BDA |
| | | Inter-I |
| | | Inter-ID |
| | | Inter-IDA |
| | | Skip |

be varied on an MB by MB basis. The MB types are listed in Table 8.2. We will take a closer look at each of these MB types in the following subsections.

6. *Blocks* are $8 \times 8$ pixel arrays. They are the smallest DCT unit. Headers are defined for sequences, GOPs, pictures, slices, and MBs to uniquely specify the data that follows. For an extensive discussion of the MPEG-1 standard, the reader is referred to [Gal 91, Pen 93].

With the introduction of B-pictures, the encoder/decoder must have sufficient memory to store at least two decoded pictures, and the "encoding/decoding order" of pictures will be different from their sequential "display order." The composition of a GOP and the concepts of "display order" vs. "encoding/decoding order" are illustrated by the following example.

### Example: Display Order vs. Encoding Order of Pictures in a GOP

A GOP of size $N=9$ and $M=4$ is shown in Figure 8.3. The first frame of each GOP is always an I-picture. The pictures are encoded in the order

$$0, 4, 1, 2, 3, 8, 5, 6, 7$$

since the prediction for P- and B-pictures should be based on pictures that are already transmitted. Picture 4 is encoded in reference to picture 0, picture 1 is encoded in reference to picture 0 and picture 4, etc.

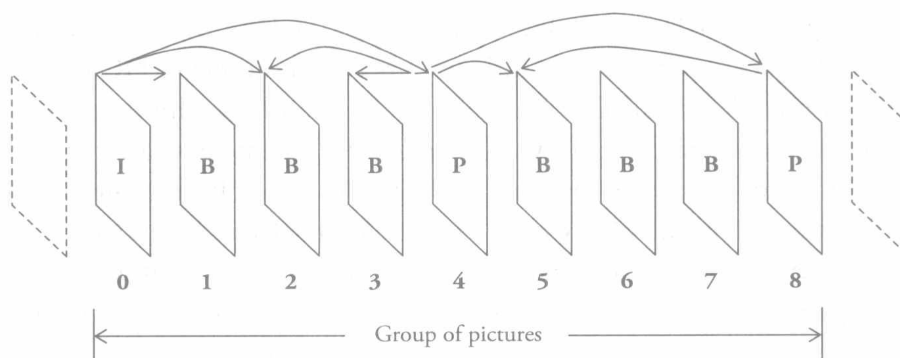



**Figure 8.3** Group of pictures in MPEG-1.

### Intra-Frame Compression Modes

In intra-coded MBs, pixel-intensity values are DCT encoded in a manner similar to JPEG or the intra-mode of H.261. Compression is achieved by quantization of the DCT coefficients and VLC of the resulting coefficients.

For 8-bit images, the DC coefficient can take values in the range [0,2047], and the AC coefficients are in the range [-1024,1023]. These coefficients are quantized with a uniform quantizer. Quantized coefficients are obtained by dividing the DCT coefficients by the step-size of the respective quantizer and then rounding the result to the nearest integer. These step-sizes are arranged into an $8 \times 8$ matrix called the *quantization matrix*. The step-size for the DC coefficient is set equal to 8 for both luma and chroma samples; thus, quantized DC values are in the range [0,255]. The AC coefficients can be represented with less than 8 bits using step-sizes larger than 8. The quantizer step-size for AC coefficients varies by frequency, according to the relative visual importance of each DCT coefficient. MPEG-1 restricts quantized AC coefficients to be in the range [-255,255].

MPEG-1 allows for spatially-adaptive quantization by introducing a quantizer scale parameter "MQUANT" in the syntax of each MB. As a result, there are two types of MBs in I-pictures: "intra" MBs are coded with the current quantization matrix. In "intra-A" MBs, the quantization matrix is scaled by MQUANT, which is transmitted in the header. MQUANT attains integer values between 1 and 31. Human visual system models suggest that MBs containing busy, textured areas can be quantized relatively coarsely. One of the primary differences between MPEG intra-mode and the original JPEG is the provision of adaptive quantization in MPEG. It has been claimed that MPEG intra-mode provides 30% better compression compared with JPEG due to adaptive quantization. Furthermore, MQUANT can be used as an effective tool for rate control.

Similar to JPEG, MPEG-1 enables prediction of the DC coefficient of each block from that of the previous block. The difference between successive DC values is VLC coded with 8 bits. The fixed-DC Huffman table has a logarithmic amplitude category structure borrowed from JPEG. Quantized AC coefficients are zigzag scanned and converted into [run, level] pairs as in JPEG. A single Huffman-like code table, which is different from that of JPEG, is used for all blocks, independent of the color component they belong to. There is no provision for downloading custom tables. Only those pairs that are highly probable are VLC coded. The rest are coded with an escape symbol followed by a fixed-length code to avoid long codewords.

**Inter-Frame Compression Modes**

In inter-frame compression modes, a temporal prediction is formed, and the resulting prediction error is DCT encoded. There are two types of temporal-prediction modes allowed in MPEG-1: forward prediction (P-pictures) and bi-directional prediction (B-pictures).

*P-pictures* allow MC forward-predictive coding with reference to a previous I- or P-picture. The forward temporal prediction $\hat{\mathbf{b}}$ for an MB $\mathbf{b}$ in the current frame $k$ is given by

$$\hat{\mathbf{b}} = \tilde{\mathbf{c}}$$

where $\tilde{\mathbf{c}}$ denotes the MB corresponding to $\mathbf{b}$ in the reconstructed previous frame as illustrated in Figure 8.4. Note that the reference picture is not necessarily the immediately previous picture.

The encoder selects the best compression mode for each MB from the list of allowable modes for a P-picture, which is shown in Table 8.2. "Intra" and "intra-A" MBs are also allowed in P-pictures, and may be selected for efficient compression of uncovered regions. MBs classified as "inter" are inter-frame coded, and the temporal prediction may or may not use motion compensation (MC) and/or adaptive quantization. The subscript "D" indicates that the prediction error will be coded, "F" indicates that forward MC is ON, and "A" indicates adaptive quantization (a new value of MQUANT is transmitted). That is, if an MB is labeled "inter-F" then the MCP $\hat{\mathbf{b}}$ is satisfactory, so it suffices to transmit just the motion vector $\mathbf{d}$ for that MB; "inter-FD" indicates that we need to transmit a motion vector and the DCT coefficients of the prediction error; and "inter-FDA" indicates that in addition to a motion vector and the DCT coefficients, a new value of MQUANT is also being transmitted for that MB. A macroblock may be "skipped" if the block at the same

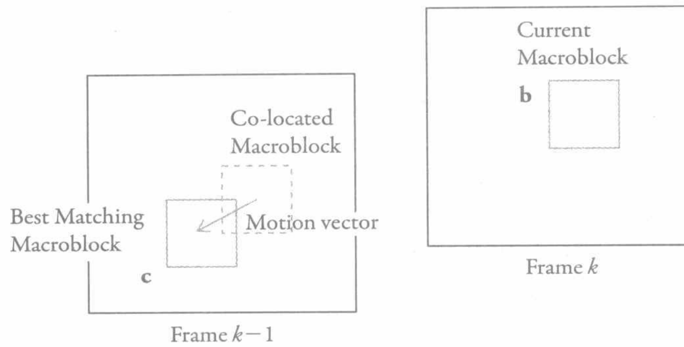



**Figure 8.4** MPEG forward prediction.

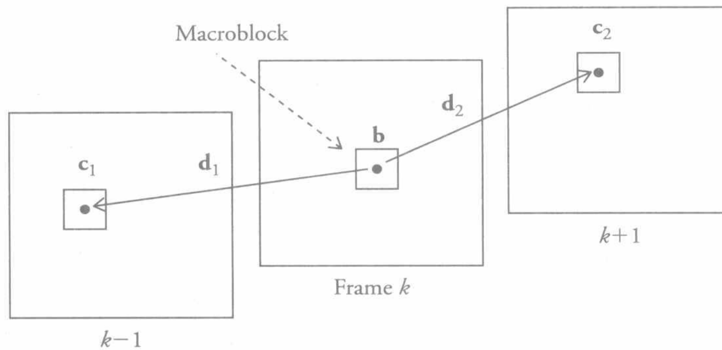



**Figure 8.5** MPEG bi-directional prediction.

position in the previous frame (without MC) is good enough, indicating a stationary "unchanged" area.

*B-pictures* allow MC interpolative coding, also known as bi-directional prediction. The temporal prediction for the B-pictures is given by

$$\hat{\mathbf{b}} = \alpha_1 \tilde{\mathbf{c}}_1 + \alpha_2 \tilde{\mathbf{c}}_2 \quad \alpha_1, \alpha_2 = 0, 0.5, 1 \text{ and } \alpha_1 + \alpha_2 = 1$$

where $\tilde{\cdot}$ denotes the "reconstructed" value. Then $\alpha_1 = 1$ and $\alpha_2 = 0$ yields forward prediction, $\alpha_1 = 0$ and $\alpha_2 = 1$ gives backward prediction, and $\alpha_1 = \alpha_2 = 0.5$ correspond to bi-directional prediction. This is illustrated in Figure 8.5. Note that in the bi-directional prediction mode, two displacement vectors $\mathbf{d}_1$ and $\mathbf{d}_2$ and the corresponding prediction error $\mathbf{b} - \hat{\mathbf{b}}$ need to be encoded for each macroblock $\mathbf{b}$.

Bi-directional prediction or interpolative coding can be considered as a temporal multi-resolution technique, where we first encode only the I- and P-pictures

(typically 1/3 of all frames). Then the remaining frames can be interpolated from the reconstructed I- and P-frames, and the resulting interpolation error is DCT encoded. The use of B-pictures provides several advantages:

- They provide an effective means for handling problems associated with covered/uncovered regions. For example, if an object is newly uncovered in the present frame, it can be non-causally predicted from the next frame.
- MC averaging over two frames may provide better SNR compared to prediction from just one frame.
- Since B-pictures cannot be used in predicting any future pictures, they can be encoded with fewer bits without causing error propagation.

The trade-offs associated with using B-pictures are:

- Two frame stores are needed at the encoder and decoder, since at least two reference (P- and/or I-) frames should be decoded first.
- If too many B-pictures are used, then the prediction distance increases resulting in lesser temporal correlation, and we have longer coding delays.

The compression mode for each MB in a B-picture is selected from the list of allowable modes shown in Table 8.2. Again, "intra" and "intra-A" MBs are allowed. MBs classified as "inter" have the following options: "D" indicates the prediction error will be coded, "F" indicates forward prediction with motion compensation, "B" indicates backward prediction with motion compensation, "I" indicates interpolated prediction with motion compensation, and "A" indicates adaptive quantization. A macroblock may be "skipped" if the co-located block from the reference frame is good enough as is; i.e., no information needs to be sent.

**Quantization and Coding**

In the inter-frame mode, the inputs to the DCT are in the range [-255,255]; thus, all DCT coefficients have the dynamic range [-2048,2047]. The quantization matrix is such that the effective quantization is relatively coarser compared to those matrices used for I-pictures. All quantized DCT coefficients, including the DC coefficient, are zigzag scanned to form [run, level] pairs, which are then coded using VLC. Displacement vectors are DPCM encoded with respect to the motion vectors of the previous blocks. VLC tables are specified for the type of MB, the differential motion vector, and the MCP error. Different Huffman tables are defined for encoding the macroblock types for P- and B-pictures, whereas the tables for motion vectors and the DCT coefficients are the same for both picture types.

### MPEG-1 Encoding

An MPEG-1 encoder includes modules for motion estimation, selection of compression mode (MTYPE), and setting the value of MQUANT at each MB, as well as MCP, quantizer and dequantizer, DCT and inverse DCT (IDCT), VLC, multiplexer, and a buffer regulator. The encoder must duplicate the decoder-processing loop so that it produces the same MC predictions as the decoder, which are obviously based on decoded previous frames. The IDCT module at the encoder should match within a prespecified tolerance, specified in the IEEE Standard 1180–1990 for 64-bit floating-point IDCT, the IDCT module at the decoder to avoid propagation of errors in the prediction process.

The number of I-, P- or B-pictures in a GOP (i.e., value of $N$ and $M$) is application-dependent. It is specified that 1 out of every 132 pictures must be an I-picture to avoid error propagation due to IDCT mismatch between the encoder and decoder. Motion vectors are represented with one-half (0.5) pixel accuracy. The maximum length of the vectors that may be represented can be changed on a picture-by-picture basis to allow flexibility. Motion vectors that refer to pixels outside the picture are not allowed. Neither the motion-estimation algorithm nor the criterion to select MYTPE and MQUANT are part of the standard. The MPEG committee has developed a simulation model encoder, called SM3, in order to verify the standard. We elaborate on some of the non-normative choices made in SM3 in the following:

1. Common choices for the number of B-frames in between two anchor frames are $M = 1$ or $M = 2$. Of course, the use of B-frames is optional in MPEG-1.
2. Motion vectors are estimated for each MB. SM3 employs logarithmic search and telescopic search methods to first find the best-integer (full-pixel) motion vector. Telescopic search is a method for reducing the search space when motion estimation is carried over multiple frames with B-frames present. To avoid large search ranges, telescopic search cascades best-motion vector estimates obtained frame-to-frame. That is, first motion vector from the current picture to the previous is searched centered about zero vector, then the best estimate from the previous picture to the next previous picture is searched centered about the best estimate from the previous step, and so on. Finally, a half-pixel update is performed about the final full-pixel motion estimate.
3. SM3 determines the compression mode MTYPE for each MB depending on the picture type of the current picture from Table 8.2. There are two approaches: i) An exhaustive method tries coding each MB using each allowed type, then chooses the one that yields the least number of bits. ii) A faster method makes a

series of decisions sequentially. In SM3, these decisions are ordered, in the case of P-pictures, as:

a. Decide whether a motion vector should be transmitted (MC) or not (No MC).
b. If no MC is selected, decide whether intra-mode or inter-mode with no motion vector will be used.
c. If inter-mode is selected, decide if the residual error is large enough to be coded (Coded) or not (Not Coded).
d. Decide if the quantizer scale is satisfactory or needs to be changed.

4. Rate control is achieved by maintaining a parametric model of the decoder, known as a video buffer verifier (VBV). The encoder monitors the status of the model buffer to update the allocation of bits to each picture type. The larger the buffer, the greater the flexibility of the encoder at the expense of larger decoding delay. The value of MQUANT is updated based on normalized spatial activity of the MB.

### 8.2.3 MPEG-2 Standard

The quality of MPEG-1 compressed video at 1.2 Mbps is unacceptable for most entertainment applications. Subjective tests have indicated that ITU-R 601 video (four times MPEG-1 SIF) can be compressed with excellent quality at 4-6 Mbps. MPEG-2 is a compatible extension of MPEG-1 for a wide range of applications at various bit-rates (4-50 Mbps) and resolutions. The main parts of the MPEG-2 standard are Systems (ISO/IEC 13818-1), Video (ISO/IEC 13818-2), Audio (ISO/IEC 13818-3), AAC Audio (ISO/IEC 13818-7), and DSM-CC (ISO/IEC 13818-6). Here, we focus on MPEG-2 video only. MPEG-2 video began as a committee draft in November 1993, and was formally approved as an international standard in 1994. It has achieved mass-market acceptance in digital TV broadcast and DVD.

The main new features of MPEG-2 video are: i) It allows efficient coding of interlaced video by introducing field pictures, frame/field adaptive MC, dualprime MC, $16 \times 8$ MC, frame/field adaptive DCT, and alternative DCT coefficient scanning options. ii) It enables higher-quality video compression (at higher bit-rates) by allowing higher-definition inputs, alternative sub-sampling of the chroma channels, and improved quantization and VLC coding options. iii) It offers a scalable bit-stream syntax option to allow for scalability and data partitioning. iv) Subsets of the full syntax have been specified under a number of "profiles." Furthermore, a number

of "levels" have been introduced within these profiles to impose constraints on some of the video parameters. In the following, we discuss the input-video formats and data structure of MPEG-2, how MPEG-2 handles interlaced video, extensions for encoding of higher-definition video, and a brief overview of the profiles and levels. More details can be found in [Gal 92] and [Chi 95].

**Input-Video Formats and Data Structure**

MPEG-2 video can take both interlaced (e.g., ITU-R 601 525/30 and 625/25) and progressive (e.g., SIF 525/30 and 625/25) inputs. MPEG-2 also allows for interlaced display of progressive coded video (e.g., progressive film source can be coded at 24 frames/s and displayed in interlaced format using 3:2 pull-down). Since it allows for 4:2:0, 4:2:2 (chroma sub-sampled only horizontally), and 4:4:4 (no sub-sampling) chroma formats, an MB in MPEG-2 may contain 6 (4 luma, 1 Cr, and 1 Cb), 8 (4 luma, 2 Cr, and 2 Cb), or 12 (4 luma, 4 Cr, and 4 Cb) $8 \times 8$ blocks. The spatial locations of luma and chroma pixels for the 4:2:0 and 4:2:2 formats are depicted in Figure 8.6.

Interlaced video is composed of a sequence of top and bottom (or even and odd) fields separated by a field period. Either the top or bottom field can be designated as the temporally first field. If the input is interlaced, the output bit-stream must consist of a sequence of fields that are separated by the field period. MPEG-2 defines two new picture types to effectively deal with interlaced video:

1. Frame pictures, which are obtained by interleaving lines of even and odd fields to form composite frames. Frame pictures can be I-, P-, or B-type. An MB of the luminance frame picture is shown in Figure 8.9(a).
2. Field pictures are simply the even and odd fields treated as separate pictures. Each field picture can be the I-, P- or B-type.

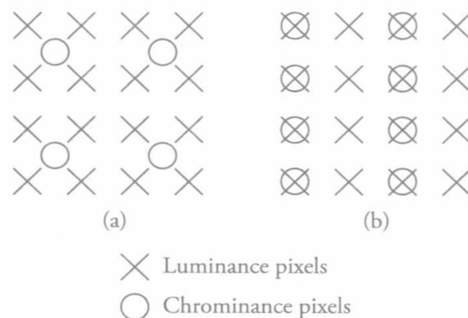



**Figure 8.6** Location of luminance and chrominance pixels for (a) 4:2:0 and (b) 4:2:2 frame pictures.

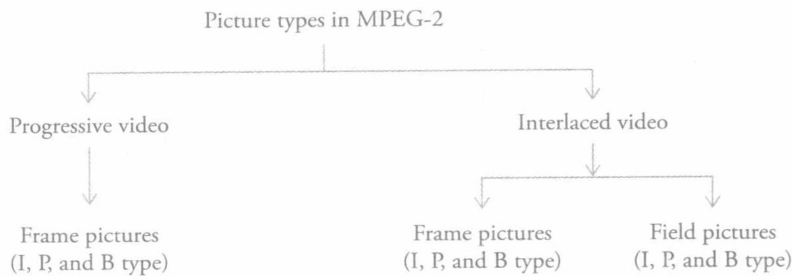



**Figure 8.7** Summary of picture types in MPEG-2.

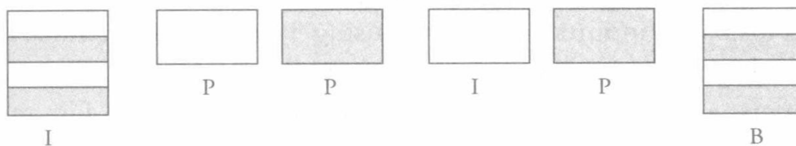



**Figure 8.8** GOP for an interlaced video.

A summary of picture types for MPEG-2 is shown in Figure 8.7. A group of pictures can be composed of an arbitrary mixture of field and frame pictures. Field pictures always appear in pairs (called the top field and bottom field), which together constitute a frame. If the top field is a P- (B-)picture, the bottom field must also be a P- (B-)picture. If the top field is an I-picture, the bottom field can be an I- or a P-picture. A pair of field pictures is encoded in the order that it should appear at the output. An example of a GOP for an interlaced video is shown in Figure 8.8. On the other hand, in progressive video all pictures are frame pictures.

### Interlaced-Video Compression

There are two options in coding interlaced video: i) every field can be encoded independently (field pictures), or ii) two fields may be encoded together as a composite frame (frame pictures). It is possible to switch between frame and field picture modes on a frame-to-frame basis. The main drawback of processing interlaced video in the form of frame pictures is that since alternate scan lines come from different fields, motion during the field period causes misalignment of spatial structure resulting in artificial vertical high-frequency content (or equivalently loss of vertical correlation). Thus, frame encoding may be preferred for relatively still images, and field encoding may give better results when there is significant motion. In order to deal with interlaced video effectively, MPEG-2 offers new options for the computation and coding of DCT coefficients and motion-compensation such as:

- New MC prediction (MCP) modes for interlaced video
- Field/frame DCT option per MB for frame pictures
- Alternate scan for ordering DCT coefficients

*MC Prediction Modes for Interlaced Video*

In order to deal with interlaced video effectively, five types of MCP are provided in MPEG-2: frame prediction for frame pictures, field prediction for field pictures, field prediction for frame pictures, $16 \times 8$ prediction for field pictures, and dual-prime prediction for P-pictures. The best prediction mode is usually frame prediction for MBs in stationary (no motion) regions and field prediction for MBs in moving regions, since in the presence of motion, frame prediction suffers from strong motion artifacts, while in the absence of motion, field prediction does not utilize all available information.

Frame prediction for frame pictures covers all MC modes that were discussed in MPEG-1 including P- and B-modes. In field prediction, each field is predicted independently using data from one or more previously decoded fields. Within a field picture only field prediction can be used. However, in a frame picture either field or frame prediction may be employed on an MB-by-MB basis. Field prediction for field pictures is similar to frame prediction except that both target MB and reference MB consist of pixels from one field. The parity of the fields for target and reference MBs may or may not be the same. To perform field prediction for frame pictures, the target MB is first split into the top and bottom fields. Field prediction is then performed independently for each of the two $16 \times 8$ parts. Thus, in this mode, two motion vectors are needed for P-pictures and two or four motion vectors are needed for B-pictures.

There are also two other prediction modes: $16 \times 8$ MC mode for field pictures and dual-prime mode for P-pictures. In $16 \times 8$ MC for field picture mode, each MB is split into an upper half and lower half, each of which is $16 \times 8$. They are motion compensated independently. Thus, two motion vectors are needed per MB, one for the upper and the other for the lower part, in P-pictures. In the case of bi-directional prediction, four motion vectors are needed. In dual-prime mode, one motion vector and a small differential vector are encoded for each MB. In the case of field pictures, two motion vectors are derived from this information and used to form predictions from two reference fields, which are averaged to form the final prediction. Dual-prime mode is used only for P-pictures [ISO 13].

*Field or Frame DCT Option for Each MB in a Frame Picture*

Prior to the computation of the DCT, the encoder may reorder the luminance lines in an MB such that the first eight lines come from the top field and the last eight

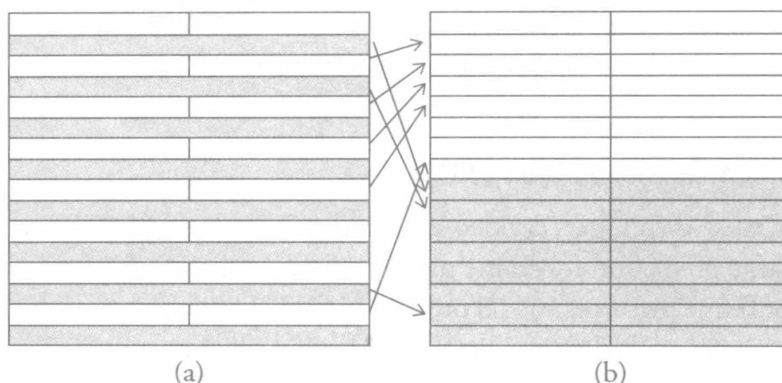



**Figure 8.9** DCT for interlaced frame pictures: (a) frame DCT and (b) field DCT.

come from the bottom field. This allows computing DCT on a field-by-field basis for specific parts of a frame picture. For example, field DCT may be chosen for macroblocks containing high motion, whereas frame DCT may be appropriate for macroblocks with little or no motion but containing high spatial activity. The internal organization of an MB for frame (a) and field (b) DCT is shown in Figure 8.9. Note that in 4:2:0 sampling only frame DCT can be used for the chroma blocks.

*Alternate Scan and Field/Frame DCT for Frame Pictures*

One way to deal with the reduced vertical resolution in frame pictures originating from interlaced video is to use a scan that favors vertical frequencies over horizontal frequencies. Therefore, MPEG-2 allows for an optional scanning pattern, called the "alternate scan," which is depicted in Figure 8.10, in addition to the zigzag scanning.

### Other Tools and Improvements

MPEG-2 features some extensions in the quantization and coding options for improved image quality in exchange for higher bit-rates. In particular, it allows for i) finer quantization of the DCT coefficients, ii) finer adjustment of the quantizer scale factor, and iii) a separate VLC table for the DCT coefficients for the intra-macroblocks, some of which are detailed in the following.

*Finer Quantization of the DCT Coefficients*

In intra-macroblocks, the quantization weight for the DC coefficient can be 8, 4, 2, or 1, i.e., 11 bits (full) resolution is allowed for the DC coefficient. Recall that this weight is fixed to 8 in MPEG-1. AC coefficients are quantized in the range

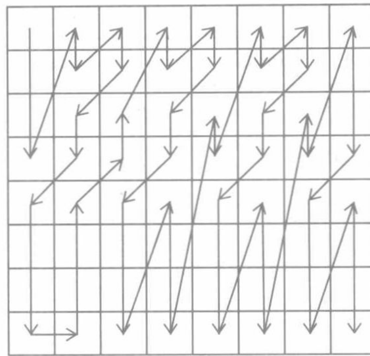


**Figure 8.10** Alternate scan.

**Table 8.3** Optional Set of MQUANT Values

| | | | | | |
|---|---|---|---|---|---|
| 0.5 | 3.5 | 9.0 | 18.0 | 32.0 | 56.0 |
| 1.0 | 4.0 | 10.0 | 20.0 | 36.0 | |
| 1.5 | 5.0 | 11.0 | 22.0 | 40.0 | |
| 2.0 | 6.0 | 12.0 | 24.0 | 44.0 | |
| 2.5 | 7.0 | 14.0 | 26.0 | 48.0 | |
| 3.0 | 8.0 | 16.0 | 28.0 | 52.0 | |

[-2048, 2047] in MPEG-2, as opposed to [-256, 255] in MPEG-1. In non-intra-macroblocks, all coefficients are quantized into the range [-2048, 2047]. This range was [-256, 255] in MPEG-1.

### *Finer Adjustment of MQUANT*

In addition to a set of MQUANT values that are integers between 1 and 31 (also known as linear quantization), MPEG-2 allows for an optional set of 31 values ranging from 0.5 to 56 (also referred as nonlinear quantization), which are listed in Table 8.3. This optional set provides higher accuracy for small coefficient values.

### *Data Partitioning*

MPEG-2 supports partitioning of a single-layer coded bit-stream into two or more layers, such that some layers are assigned a higher priority (hence, quality-of-service parameters) for reliable transmission over heterogeneous networks. Data partitioning may be considered as an elementary form of scalable video representation.

**Table 8.4** Parameter Constraints According to Levels

| Level | Max. Pixels | Max. Lines | Max. Frames |
|---|---|---|---|
| LOW | 352 | 288 | 30 |
| MAIN | 720 | 576 | 30 |
| HIGH-1440 | 1440 | 1152 | 60 |
| HIGH | 1920 | 1152 | 60 |

### Profiles and Levels

MPEG-2 full syntax covers a wide range of features and free parameters. Considering practical difficulties with the hardware implementation of the full syntax, there are six MPEG-2 profiles that define subsets of the syntax and four levels that impose constraints on the values of the free parameters. The profiles are the Simple, Main, SNR, Spatial, High, and 4:2:2 profiles. The parameter constraints imposed by the four levels are summarized in Table 8.4.

Simple, Main, SNR, Spatial, and High profiles are designed in an "onion-ring" structure, i.e., the High profile supports all tools supported by the previous four profiles and some new ones, the Spatial profile supports all tools supported by the previous three and new ones, and so on. The Simple profile does not allow B-pictures and supports only the Main level. The Main profile does not include any scalability tools (see Section 8.5) and supports all four levels with upper bounds on the bit-rates equal to 4, 15, 60, and 80 Mbps for the Low, Main, High-1440, and High levels, respectively. The High profile is a superset of the Spatial profile such that it also supports 4:2:2 video. The 4:2:2 profile addresses professional digital-video applications, which requires 4:2:2 chroma sampling but not scalability.

The Main profile at the Main level (MP@ML), which is used for standard-resolution digital TV, has been by far the most widely adopted MPEG-2 profile. The 4:2:2 profile at the Main level is required to decode all bit-streams decodable by MP@ML decoders.

### MPEG-2 Encoding

MPEG-2 encoding is more complex than MPEG-1 encoding since the encoder needs to consider several more encoding mode choices to obtain the best results. We discuss some non-normative elements of the MPEG-2 Test Model 5 (TM5) encoder developed by the MPEG group in order to verify the standard. TM5 estimates two types of motion vectors (MVs), frame MVs and field MVs, for each MB of P- and B-frame pictures. For B-frames, two frame-motion vectors (one forward and one

backward) are estimated for each $16 \times 16$ MB. In addition, four field motion vectors (two for each direction) are estimated, where each MV corresponds to a $16 \times 8$ luminance region. For P-frames, only forward frame- and field-motion vectors (total of 3) are estimated. Field MVs are estimated to minimize the sum of absolute errors in the respective fields, where the frame vectors minimize the sum of the errors in the two fields.

In order to decide for the best compression mode, the following procedures are performed at each MB: For P-pictures, TM5 first decides between a frame MV vs. two field MVs based on comparison of the respective sum of absolute errors. For B-pictures, there are a total of six possible combinations, which are combinations of forward/backward/interpolated and frame/field MVs, at each MB. TM5 computes the sum of absolute errors for each combination and chooses the one that yields the minimum value. Then, a decision must be made between transmission of the selected motion vector(s) (MC) vs. not transmitting any MV (no MC). This is also based on comparison of the respective sum of absolute errors. Next, a decision is made between intra- vs. inter-coding. This decision criterion, based on comparison of variances of the block difference and MC block difference, is the same as in MPEG-1 SM3. After the intra/inter decision, TM5 decides for frame vs. field DCT. To this effect, each $16 \times 16$ luminance MB is rearranged as four $8 \times 8$ field blocks, where each block contains pixels only from a single field. TM5 then computes the vertical correlation of the original (frame) and rearranged (field) block configurations and chooses the one that gives higher correlation as the DCT type.

An MPEG-video decoder de-multiplexes the incoming video bit-stream (with a standard syntax) into image data and side information such as MTYPE, motion vectors, MQUANT, and so on. The inverses of the encoder operations are performed on the image data. The decoder must employ at least two frame-stores, since two reference frames are needed to decode B-pictures.

## 8.3 MPEG-4 AVC/ITU-T H.264 Standard

ITU-T H.264/MPEG-4 Part 10 Advanced Video Coding (AVC) is a state-of-the-art video coding standard developed by the Joint Video Team (JVT) consisting of experts from ITU-T VCEG and ISO/IEC MPEG [Wie 03, Sul 05]. It improves the coding efficiency (on average by a factor of two) over MPEG-2 at the cost of some increase in complexity. Fidelity range extensions (FRExt) were added as an amendment to the standard in July 2004 to support the requirements of professional high-fidelity video applications. FRExt has provided further coding efficiency improvements (up to a

factor of 3) over MPEG-2 for high-fidelity video. H.264/AVC introduces several innovative concepts and tools, which are discussed in the following.

The original version of the standard specifies three profiles: Baseline (low complexity and robustness), Main (high compression efficiency), and Extended (high compression efficiency and robustness). A family of *High profiles*, which consists of the High profile (HP), High 10 profile (Hi10P), High 4:2:2 profile (Hi422P), and High 4:4:4 profile (Hi444P), were added with the FRExt amendment and support adaptive transform block size and perceptual quantization scaling matrices in addition to all features of the previous Main profile. The Hi444P also supports an integer residual color transform for coding RGB video [Sul 05].

## 8.3.1 Input-Video Formats and Data Structure

The Baseline profile accepts progressive video with 4:2:0 color sampling and 8-bit per sample, per component accuracy. The Main, Extended, and High profiles accept progressive or interlaced video with 4:2:0 color and 8-bit per sample/component accuracy. Hi10P allows for 4:2:0 color with a 10-bit per sample/component. Hi422P allows for 4:2:2 color with 10-bit per sample/component, while Hi444P allows for 4:4:4 color with 12-bit per sample/component sample accuracy.

The data structure of H.264/AVC is organized as a sequence of pictures, which consist of slices, which in turn consist of macroblocks, that can be expressed as

$$\text{Sequence(pictures(slices(macroblocks)))}$$

Each sequence starts with an instantaneous decoding refresh (IDR) access unit; i.e., a picture that can be decoded without decoding any previous pictures. An IDR picture indicates that no subsequent picture in the stream will require reference to pictures prior to the IDR picture.

### Macroblocks and Slices

A macroblock (MB) is defined as a $16 \times 16$ luma block and the corresponding chroma blocks as usual. MBs are grouped into slices. A picture may contain one or more slices that are independently decodable. Slices i) help generation of payload packets that can fit the maximum-transfer unit (MTU) of a network so that each MTU carries an integer number of slices, ii) help error resilience in lossy transmission environments so that a lost packet affects only a limited region of a picture, and iii) enable parallel encoding and decoding since each slice can be independently encoded/decoded. Picture types are not used in H.264/AVC. Instead, there are five slice types: I-slice (all MBs are coded by using intra-prediction only), P-slice (up to one motion vector per sub-block),

B-slice (up to two motion vectors per sub-block), switching P (SP) slice, and switching I (SI) slice. That is, a picture may contain slices of different types. SP and SI slices are included in the Extended profile to facilitate switching between different bit-streams representing the same video encoded at multiple quality levels (multiple bit-rates).

**Temporal-Prediction Structures and Processing Order**

The classical temporal-prediction structures are "IPPPPP..." picture ordering with sequential encoding/decoding or "IBBBPBBBP..." picture display ordering with the corresponding encoding/decoding order shown in the Example in Section 8.2.2. H.264/AVC allows for more flexible slice dependency and temporal-prediction structures. For example, a picture can be marked as a reference picture regardless of the coding types of its slices, stored in the decoded picture buffer (DPB) that can hold up to 16 pictures, and used for MCP of future pictures before the next IDR picture.

Experiments have indicated that hierarchical prediction structures that enable multi-level temporal scalability of the bit-stream also increase the coding efficiency. An example of a hierarchical prediction structure with four dyadic levels is depicted in Figure 8.11, where pictures at the highest level are called key pictures. Key pictures are either intra-coded (to enable random access) or inter-coded using previous key pictures as reference for MCP. In the example, the first picture with the picture order count (POC) 0 and the picture with POC 8 are key pictures. Only the first picture is intra-coded as an IDR picture. The remaining pictures are hierarchically predicted as illustrated, such that the second-level picture with POC 4 is predicted from the two key pictures, and the third-level pictures with POC 2 and 6 are predicted as shown by the arrows. The fourth-level pictures are marked "b," which implies they are not a reference for other pictures.

## 8.3.2 Intra-Prediction

In MPEG 1/2, similar to JPEG, intra-prediction has been limited to prediction of the DC coefficients across blocks in the frequency domain. H.264/AVC employs a novel approach to block-based intra-prediction in the spatial domain. All slice

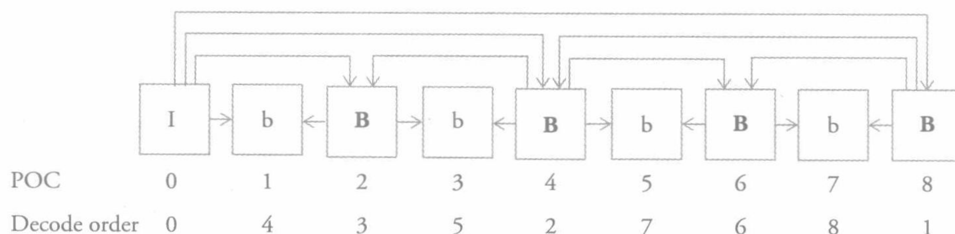



**Figure 8.11** Hierarchical B-pictures. POC denotes the display order.

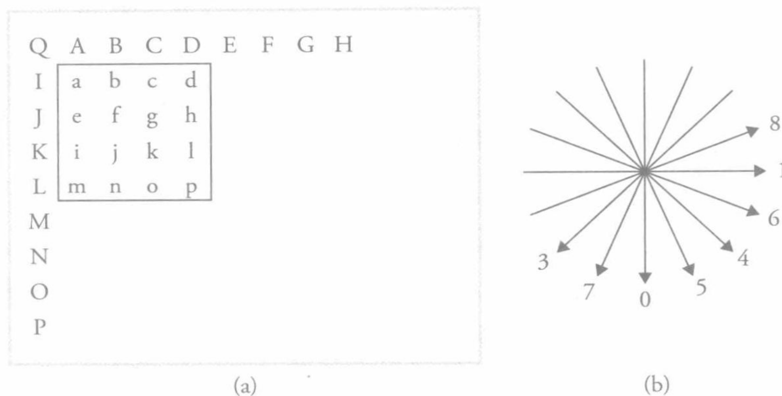



**Figure 8.12** Illustration of 4 × 4 intra-prediction: (a) 4 × 4 block and boundary pixels; (b) prediction directions.

types in the Baseline, Main, and Extended profiles support two types of luma intra-prediction: 4 × 4 and 16 × 16 intra-prediction. In addition, High profiles (FRExt) also support 8 × 8 intra-prediction. The chroma samples are predicted by using the modes in the 16 × 16 intra-prediction.

In 4 × 4 prediction, 16 samples of a 4 × 4 block denoted by small letters a through p are predicted from border samples of previously encoded MBs marked by capital letters A through Q in Figure 8.12(a). The encoder can select either the DC mode (mode 2), where all samples a through p are predicted by an average value, or one of modes 0–1,3–8, which correspond to prediction in the directions indicated by the arrows in Figure 8.12(b), e.g., in mode 0, a = e = i = m = A, b = f = j = n = B, c = g = k = o = C, and d = h = l = p = D; in mode 1, a = b = c = d = I, e = f = g = h = J, i = j = k = l = K, and m = n = o = p = L.

In 16 × 16 prediction, only four modes are supported: DC, horizontal, vertical, and planar. The first three modes are the same as in 4 × 4 prediction, except they extend to the entire 16 × 16 block. Planar mode predicts the current block by a plane-fit approximation to model the horizontal and vertical variation in the intensities of the border pixels of neighbor blocks.

## 8.3.3 Motion Compensation

Over the years, significant gains in compression ratios have been achieved by advances made in the motion-compensated prediction (MCP) methods. In MPEG 1/2, we have P- and B-MCP modes with 16 × 16 fixed block size and half-pixel accuracy for motion vectors (MV). H.264/AVC brings several innovations for improved MCP.

They include: quarter-pixel accuracy (1/8 pixel accuracy for chroma) MCP, reference picture extrapolation to handle MVs extending outside picture boundaries, variable size blocks for MCP, multi-picture MCP, and multi-hypothesis and weighted MCP, which are explained below.

### Motion-Vector Precision and Encoding

The precision of MVs is 1/4 of the distance between luma pixels. The corresponding sub-pixel sample values are computed by interpolation. Half-pixel sample values are evaluated by using a separable 6-tap FIR filter horizontally and vertically, while quarter-pixel sample values are computed by bi-linear interpolation between full- and computed half-pixel samples. Chroma samples are always computed by bi-linear interpolation. If the MVs point to outside of reference picture boundaries, the reference picture is extrapolated by replicating border pixels. The components of the MV are encoded predictively using either median or directional prediction from the MVs of neighboring blocks. No prediction takes place across slice boundaries.

### Variable-Block-Size MCP

Each luma MB in P- and B-slices can be partitioned into blocks of different sizes and shapes for MCP. H.264/AVC syntax supports $16 \times 16$, $16 \times 8$, $8 \times 16$, and $8 \times 8$ blocks as shown in Figure 8.13. In addition, each $8 \times 8$ block can be further subdivided into $8 \times 4$, $4 \times 8$, or $4 \times 4$ sub-blocks. Thus, a maximum of 16 MVs may need to be transmitted for each MB in P-mode MCP.

### Multi-Picture MCP

Multi-picture MCP refers to using more than one previously coded picture as unidirectional references even in P-slices, as depicted in Figure 8.14. It requires both the

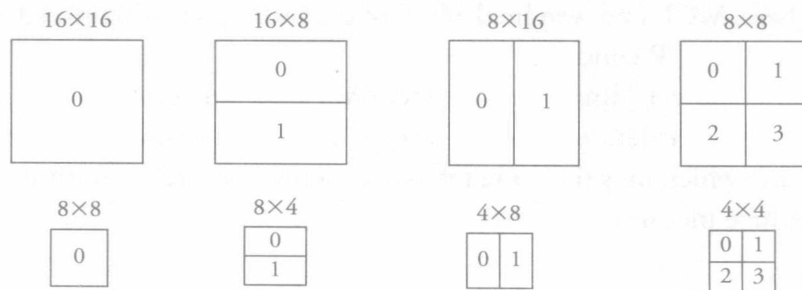



**Figure 8.13** Sub-blocks for motion compensation.

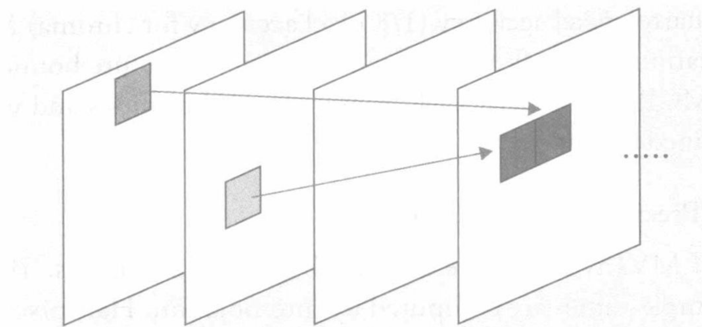


**Figure 8.14** Illustration of multiple-picture motion-compensated P-prediction.

encoder and decoder to store the same pictures in their multi-picture buffers. This is achieved by memory-management control-operations (MMCO) messages that are sent in the bit-stream, where the encoder signals the index of the reference picture for each $16 \times 16$, $16 \times 8$, $8 \times 16$, and $8 \times 8$ block. Sub-blocks smaller than $8 \times 8$ use the same reference picture.

**Multi-Hypothesis MCP, Weighted MCP, and B-Slices**

With the introduction of multi-picture MCP and allowing B-slices to be used as reference for other slices, the main difference between P-slices and B-slices in H.264/AVC becomes that B-slices allow weighted bi-prediction. In previous standards, bi-directional weighted prediction was performed by a simple (1/2, 1/2) averaging filter. In H.264/AVC, the encoder can specify weights and offsets to be used in each P- and B-macroblock of a slice. The encoder is allowed to specify different weights and offsets within the same slice. This can be especially effective in encoding "cross fades" as B-slices allow weighted blending between pictures from two scenes (shots). Offsets allow multi-hypothesis MCP, i.e., linear superposition of MCPs, such as in overlapped block-motion compensation. Indeed, combining multi-picture MCP, multi-hypothesis MCP, and weighted MCP offers a very powerful unified generalization of all known MCP concepts.

B-slices also allow a "direct mode" where MVs for a macroblock are not explicitly sent. The decoder derives MVs by scaling MVs of a co-located macroblock in another picture, which uses the same reference picture, according to time differences between the three pictures.

## 8.3.4 Transform

H.264/AVC uses $4 \times 4$ transform blocks and a separable integer transform that closely approximates DCT instead of the DCT. The coefficients of $4 \times 4$ DCT vs.

$$T=\begin{bmatrix} .50 & .50 & .50 & .50 \\ .65 & .27 & -.27 & -.65 \\ .50 & -.50 & -.50 & .50 \\ .27 & -.65 & .65 & -.27 \end{bmatrix}$$

(a)

$$T=\begin{matrix} 1/2 \\ 1/\sqrt{10} \\ 1/2 \\ 1/\sqrt{10} \end{matrix}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix}$$

(b)

**Figure 8.15** Comparison of 4 × 4 transforms: (a) DCT and (b) integer transform in H.264/AVC.

the integer transform are shown in Figure 8.15. FRExt also supports an 8 × 8 integer transform. Since the inverse transform is an integer transform, rounding errors do not occur.

We note that since H.264/AVC has improved prediction modes, the intra- and inter-prediction error (residual) has less spatial correlation; hence, the 4 × 4 transform is as efficient as the usual larger 8 × 8 transform in removing correlations. The smaller transform has the following benefits: i) it visually results in less mosquito noise and ringing artifacts around edges, ii) it can be implemented by using only adds and shifts and, hence, avoids the encoder-decoder mismatch problem, and iii) it requires only 16-bits wordlength for all operations including scaling.

The DC coefficients of the luma intra-16 × 16 mode and all chroma modes undergo a second transformation. This Haar transformation is 4 × 4 for the intra-16 × 16 mode and 2 × 2 for all others. The second transformation aims to exploit the remaining redundancy between DC coefficients of the 4 × 4 blocks, which proves useful in relatively flat image areas.

## 8.3.5 Other Tools and Improvements

### Quantization

A uniform quantizer whose step-size is controlled by a quantization parameter (QP) that can take on 52 different values is used. The quantization step-size changes logarithmically by QP, and an increase by one in QP corresponds to an approximately 12% increase in the step-size.

### Entropy Coding

There are two entropy coders in H.264/AVC: context-adaptive variable-length coding (CAVLC) and context-adaptive binary arithmetic coding (CABAC). CABAC has higher complexity and is supported in Main and High profiles only.

In CAVLC, the quantized transform coefficients and motion vectors are coded using VLC tables that are context conditional, i.e., switched according to the values

of the previous syntax elements. Different context models are used for motion vectors and transform coefficients. Instead of designing a different VLC table for each context, only the mapping from a single universal code table, refered to as Exp–Golomb code, to each context is performed.

In CABAC, after context estimation, symbols are mapped into a sequence of binary decisions, where each decision is encoded by binary arithmetic coding. CABAC not only uses context-conditional probability estimates, but also encodes each symbol with a non-integer number of bits, as we have learned when studying arithmetic coding. Compared to CAVLC, CABAC improves coding efficiency by 10% to 12% at the expense of increased computational complexity.

### In-Loop De-Blocking Filter

Various de-blocking filters have been developed for post-processing of decompressed frames to reduce blocking artifacts resulting from processing each block independently in JPEG and MPEG-1/MPEG-2. In H.264/AVC, the de-blocking filter is used in the encoding-decoding loop; hence, it is a normative part of the standard. The filter strength can be selected *a priori* or determined by the encoder according to coding modes of adjacent blocks, quantization step-size, and steepness of the luminance gradient between blocks. The filter operates on the edges of each $4 \times 4$ or $8 \times 8$ transform block. It can modify up to three pixels on either side of a given block edge depending on the filter-strength value. The filter can also be turned off by the encoder.

### Error Resilience Tools

Partitioning pictures into slices helps localize the adverse effects of network packet losses since the start of each slice is a resynchronization point at the decoder. In addition, H.264/AVC provides special tools to enhance error resilience of the bit-stream, including data partitioning, flexible macroblock ordering (FMO), arbitrary slice ordering (ASO), and redundant slices. Data partitioning refers to separating more important data such as MB types and MVs from less important data such as transform coefficient values and sending them in separate packets. FMO refers to non-sequential mapping of MBs to different slices, such as by a checkerboard pattern. ASO refers to non-sequential ordering of slices of a picture in the bit-stream. Redundant slices allow sending duplicative coded representations of some or all parts of a picture.

The coded data is organized into network-access layer (NAL) units (also called packets) containing an integer number of bytes. The first byte of each NAL unit is a header, and the remaining bytes are the payload data of the type indicated by the

header. When streaming over the Internet, the NAL units are encapsulated by underlying transport protocol packets.

An open-source implementation of the AVC encoder and decoder, known as x264 library, was released under the GNU GPL license and is available at: http://www.videolan.org/developers/x264.html.

## 8.4 High-Efficiency Video-Coding (HEVC) Standard

The high-efficiency video-coding standard (HEVC) is the most recent video-compression standard jointly developed by ITU-T VCEG and ISO/IEC MPEG and its first version was published in 2013 [Sul 12]. The ITU term for HEVC is H.265, while the MPEG name is MPEG-H Part 2. It is designed to provide an average of 50% improvement in bit-rate for ultra-high-definition video compared to H.264/AVC at the same visual quality by aggregating a number of small improvements. The basic HEVC design is still based on the classical MC transform coding, and also includes a video-coding layer (VCL) that refers to actual video, and a network-abstraction layer (NAL) that refers to transport interface aspects. The main technical novelties are replacing the concept of MB with coding-tree units (CTU) and introducing tools for parallel video encoding/decoding. Scalable coding and 3D-video extensions of the standard are in progress.

### 8.4.1 Video-Input Format and Data Structure

HEVC assumes the input is progressive video since most cameras and displays are now progressive, and interlaced video is becoming less common. That is, HEVC includes no explicit tools for efficient encoding of interlaced video in order not to burden decoders with additional complexity, i.e., no more macroblock-adaptive frame-field (MBAFF) coding. However, metadata syntax is provided to signal an interlaced-video input where each coded picture can be a separate field or a composite interlaced frame.

HEVC high-level syntax includes new features for random access and bit-stream splicing. In H.264/AVC, a conforming bit-stream must start with an *instantaneous decoder refresh* (IDR) picture that defines a closed GOP. HEVC introduces a distinct NAL unit type to signal a *clean random-access* (CRA) picture, which enables an open GOP. That is, an HEVC-conforming bit-stream may start with an IDR or a CRA picture. In a closed GOP, all pictures are decodable without referencing pictures from other GOPs. In an open GOP, some pictures can depend on pictures preceeding the CRA picture. Pictures preceeding a CRA picture in display order, but

appearing after the CRA picture in decoding order, are called *leading pictures*. If a leading picture references a picture preceeding the CRA picture, it should be skipped in a decoding process that starts from the CRA picture, and it is called a *random access skipped leading* (RASL) picture. If a leading picture does not contain any reference to pictures preceeding the CRA picture, it is called a *random-access decodable leading* (RADL) picture. Thus, CRA pictures provide random-access points (RAP) without flushing the decoded picture buffer (DPB); i.e., without breaking temporal dependencies. Open GOPs are sometimes preferred because they generally provide better compression efficiency compared with closed GOPs. A broken-link access (BLA) picture is a special CRA picture used to signal a bit-stream splicing point. To summarize, in HEVC, random access is provided at IDR, CRA, and BLA pictures.

## 8.4.2 Coding-Tree Units

In HEVC, the concept of macroblock, which was the basic coding unit in the previous standards, was replaced by coding-tree units (CTU), such that each slice of a picture is partitioned into CTUs consisting of a coding-tree block (CTB) of luma pixels and two coding-tree blocks of corresponding chroma-pixels. CTUs can be $64 \times 64$, $32 \times 32$, or $16 \times 16$. The width and height of a CTU are signaled in a sequence parameter set, so that all CTUs in a video sequence have the same size. With the introduction of ultra-high-definition video formats, we see that larger block sizes for MCP and/or transform provide higher compression efficiency. In Class A test sequences with resolution $2560 \times 1600$, it was found that the bit rate increases by 5.7% when forced to use $32 \times 32$ CTU and increases by 28.2% when forced to use $16 \times 16$ CTU compared to $64 \times 64$ CTU [Ohm 12]. Large CTU sizes also reduce the decoding time.

Each CTB can be differently split into flexible-square coding blocks (CB). HEVC supports CB sizes from the same size as the CTB to as small as $8 \times 8$. The partitioning of each CTB into CBs is conveyed using hierarchical quad-tree syntax. Partitioning of a $64 \times 64$ CTB into CBs and its quad-tree representation are illustrated in Figure 8.16, where some CBs are $32 \times 32$ and others are $16 \times 16$ and $8 \times 8$. A coding unit (CU) consists of a Y and associated Cr and Cb CBs with their syntax elements. The prediction-type decisions are made and signaled at the CU level.

Each CB can be further split into prediction blocks (PB) depending on their temporal and/or spatial predictability. In intra-prediction, the PB size is set the same as the CB size except for the smallest CB size in the bit-stream, which can be further split into four quadrants. As a result, different intra-prediction modes can be selected for PBs as small as $4 \times 4$. In inter-prediction, luma and chroma CBs can be further split into two or four PBs. Splitting into four is allowed only with the smallest CB

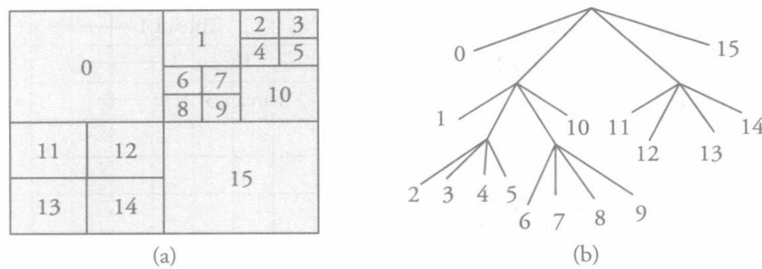



**Figure 8.16** Partitioning a coding-tree block (CTB) into coding blocks (CB): (a) an example partition and (b) associated quad-tree representation.

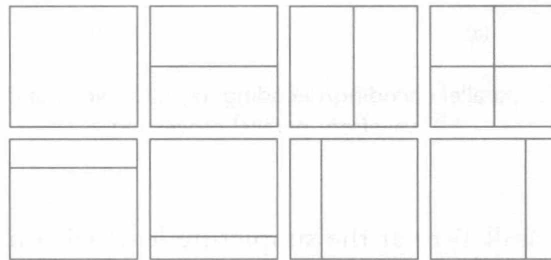


**Figure 8.17** Partitioning of coding blocks (CB) into prediction blocks.

size. Partitioning of a CB into PBs is illustrated in Figure 8.17, where the asymmetric splits with a ratio of 1/4 and 3/4, depicted in the lower row, are only allowed for CBs $16 \times 16$ or larger. Each PB can be assigned one or two MVs. To avoid a large number of MVs, $4 \times 4$ luma PBs are not allowed in inter-prediction, and $4 \times 8$ and $8 \times 4$ luma PBs can only be used in unidirectional inter-prediction.

Independent of partitioning into PBs, each CB can also be split into transform blocks (TB) whose boundaries need not be aligned with PB boundaries. That is, it is possible to perform a single transform across residuals from multiple PBs for inter-prediction CUs. Only square TBs can be specified and can be as small as $4 \times 4$. The TB partitioning information is signaled by a residual quad-tree.

## 8.4.3 Tools for Parallel Encoding/Decoding

HEVC supports special tools, such as tiles and wavefront parallel processing, that are designed to enable decoding a single picture with multiple threads.

### Tiles

Tiles are independently decodable rectangular regions of an image, similar to those defined in the JPEG2000 standard. Tiles should be at least $256 \times 64$ luma

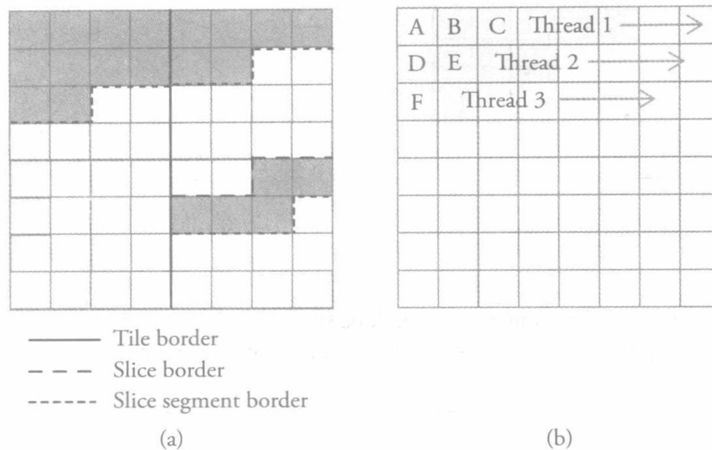



**Figure 8.18** Tools for parallel encoding/decoding: (a) tiles, slices, and slice segments and (b) wavefront parallel processing.

pixels. They enable parallelism at the subpicture level with no need for synchronization between threads. They also facilitate region-of-interest (ROI) decoding. Multiple tiles may be contained within a single slice sharing a common header, or alternatively a tile may contain multiple slices, including dependent slice segments [Mis 13].

Dependent slice segments are defined to adjust the granurality of packetization to assist low-delay encoding. With dependent slice segments, data associated with particular tiles or wavefront entry points can be carried in separate NAL units (packets) with lower latency than if they are all packetized in a single slice. However, they are not independently decodable, since they do not contain slice headers.

The relationship between tiles, slices, slice segments, and CTUs is illustrated in Figure 8.18(a), where a picture consisting of $8 \times 8$ CTUs is divided into two tiles denoted by the vertical solid line. Note that tiles are always aligned with CTU boundaries. The first tile contains a single slice, which is divided into two slice segments, whose boundaries are denoted by dotted lines. The first slice segment, marked by shaded CTUs, is an independent slice segment and the other is configured as a dependent slice segment. The second tile is split into two slices, each with one independent and one dependent slice segment.

### Wavefront Parallel Processing

Wavefront processing enables subslice-level parallelism by partitioning slices into rows of CTUs. Each row can be processed by an independent thread, as depicted in

Figure 8.18(b), but there should be two-CTU processing delay between the threads. That is, processing of the second row can start only after the first two CTUs from the first row (A and B) have been completed, i.e., C and D can be processed in parallel, and processing of the third row can start after two CTUs from the second row (D and E) have been completed, and so on. Wavefront parallelism cannot be used with tiles and it often provides better compression efficiency compared to tiles, since intra-prediction or prediction of motion vectors cannot be performed across tile boundaries.

### 8.4.4 Other Tools and Improvements

HEVC features a number of small improvements that yield large coding gains when combined together.

**Intra-Prediction**

In intra-prediction all PBs are square with the same size as TBs, i.e., $32 \times 32$, $16 \times 16$, $8 \times 8$, or $4 \times 4$. Intra-prediction partition follows the TB quad-tree. The intra-prediction in HEVC supports DC, planar, and angular prediction modes for all TB sizes and slice types. In the DC mode, all samples in a block are set equal to the mean value of boundary samples. Planar mode predicts all samples in a block by fitting a planar amplitude surface, where horizontal and vertical slopes are computed from the boundary samples. In the angular prediction mode, HEVC defines 33 prediction directions as opposed to 8 directions in H.264/AVC. The projected sample locations are computed with 1/32 sample accuracy using bi-linear interpolation of samples with the closest integer locations. For improved accuracy, reference-sample smoothing, boundary-value smoothing, and reference-sample substitution procedures may be employed in HEVC.

**Motion Compensation**

Inter-prediction mode allows symmetric and asymmetric CB partitions, as shown in Figure 8.17. HEVC specifies quarter-sample accuracy for luma motion vectors, similar to AVC, but uses separable 8-tap and 7-tap interpolation filters for half- and quarter-sample luma positions, respectively (as opposed to 6-tap and 2-tap bi-linear for half- and quarter-sample positions, respectively, in AVC) and 1/8 sample motion vector accuracy and 4-tap filter for chroma. All PB blocks need to be extended on all sides to provide the filter with the required boundary samples. HEVC supports weighted prediction for both uni- and bi-directional PBs, where the weights are explicitly transmitted in the slice header, and there is no implicit weighted prediction

as in AVC. Similar to AVC, HEVC has two reference lists, L0 and L1, that can hold 16 references each, but the maximum number of unique pictures is 8. The encoder may choose to add the same picture to the list more than once to enable predicting the same picture with different weights.

### Motion-Vector Coding

There are two MV prediction modes: merge and advanced-motion vector prediction (AMVP). The encoder decides between these two modes for each prediction unit (PU) and signals it in the bit-stream. The merge process is a generalization of the direct mode in AVC, except that it explicitly states the reference picture list and index, whereas in the direct mode they take implicit values. When an inter-predicted CB is not encoded in skip or merge modes, AMVP is employed to encode a delta MV. Both merge and AMVP build a list of candidate MVs and then select one of them using an index coded in the bit-stream.

### Transform and Quantization

The core transform in HEVC is a separable $32 \times 32$ integer transform that approximates the DCT. The coefficients of the $16 \times 16$, $8 \times 8$, and $4 \times 4$ transforms are derived from this by sub-sampling. There is also an alternate $4 \times 4$ integer transform that approximates the discrete-sine transform (DST), which is applied to luma residuals in intra-predictive coding.

HEVC uses the same uniform reconstruction quantization (URQ) scheme controlled by a QP as in the AVC. The QP values range from 0 to 51. There is an approximate logarithmic mapping between QP values and URQ step-sizes, where increasing QP by 6 corresponds to doubling the quantization step-size. HEVC also supports quantization-scaling matrices.

### Entropy Coding

HEVC employs only CABAC for entropy coding. The core algorithm is the same as that in AVC; however, improvements are made in context modeling, adaptive coefficient scan, and coefficient coding for better compression efficiency. There are about half as many context-state variables in HEVC compared with AVC, and the initialization is simpler. CABAC decoding is inherently a serial operation, and fast/multi-thread hardware implementations of CABAC are difficult. Careful use of the by-pass mode and dependencies between coded data have been successfully exploited for throughput maximization of hardware decoders and multithread implementations.

### In-Loop De-Blocking Filter

De-blocking in HEVC is performed on $8 \times 8$ blocks only, unlike AVC where it is applied to $4 \times 4$ block edges. All vertical edges in the picture are de-blocked first, followed by all horizontal edges. The filter is similar to that in AVC, but only boundary strengths 2, 1, and 0 are supported. The 8-pixel separation between de-blocking filters enables parallel implementation, since edges do not depend on each other. Hence, it is possible to perform vertical edge filtering with one thread for each 8-pixel column of the picture. Chroma is de-blocked only when one of the PUs on either side of a particular edge is intra-coded.

Profiles, tiers, and levels define conformance points for implementing the standard. Tiers define limits on the maximum bit-rate and coded picture buffer (CPB). HEVC currently defines three profiles, Main, Main10, and Main Still Picture, and two tiers, Main and High. Work is on-going on future extensions of HEVC including coding of extended range formats (bit-depth, color sampling), scalable video coding, and stereo/3D-video coding.

An open-source implementation of HEVC encoders and decoders, known as the x265 application library, is available under the GNU GPL 2 license (http://x265.org/).

## 8.5 Scalable-Video Compression

Scalable-video coding (SVC; also known as layered-video coding) refers to the general approach to video encoding where subsets of the bit-stream can be decoded to generate complete video sequences, whose spatial/temporal resolution and/or quality vary according to the selected subsets. SVC also refers to the specific encoding methodology described in the Annex G extension of the H.264/AVC standard. The minimum decodable subset of the bit-stream is called the base layer. All other layers are enhancement layers that improve the resolution or quality of the base-layer video. In SVC, the base layer is encoded independent of enhancement layers. However, each enhancement layer is coded in reference to a previous lower level. This increases the compression efficiency of the scalable-video stream compared to simulcasting (which refers to coding each layer independently).

Scalable video enables serving video to users with different displays (formats), bit-rate, and power requirements from a single stream, and allows decoders of different complexities to coexist. While low-performance decoders process smaller subsets of the bit-stream producing basic quality video, high-performance decoders may decode larger subsets to produce higher-quality video. Scalable video also enables

easy network adaptation for robust transmission over heterogeneous networks. In case of congestion or lower bandwidth connections, enhancement layers can be dropped, allowing the decoder to at least receive the base layer in its entirety.

Temporal, spatial, and quality scalable video-coding tools were first introduced in the MPEG-2 specification, which allows for two or three layers of video. However, MPEG-2 spatial and quality scalability features have resulted in a noticeable loss of compression efficiency while increasing encoder/decoder complexity. Wavelet-based approaches for SVC have also been studied but have not been found as efficient as the scalable extension of the H.264/AVC standard. Most recently, the HEVC standard was extended to include scalable coding features, called the SHVC, and was completed in July 2014. SHVC supports parallel encoding and decoding of ultra-high-definition videos [Ham 14], where the base layer can be an AVC bit-stream. This section only introduces the basic tools to provide temporal, spatial, and quality scalable bit-streams in the scalable extension of the H.264/AVC standard, which successfully addresses compression efficiency and complexity problems.

## 8.5.1 Temporal Scalability

*Temporal scalability* refers to the ability to decode video at different frame rates. For example, in MPEG-2, since B-pictures cannot be used as reference, they can be dropped without affecting decodability of I- and P-pictures, providing a limited temporal scalability. In H.264/AVC, temporal scalability is provided with more flexibility by introducing hierarchical prediction structures by only adding syntax to the H.264/AVC design to signal temporal layers.

A dyadic and a low-delay structure for hierarchical prediction are illustrated in Figure 8.19(a) and (b), respectively. In the dyadic structure using only B-pictures (also called hierarchical B-pictures; see Figure 8.11), all temporal layers $T_l$, $l > 0$, where $T_0$ denotes the base layer, can be decoded independently by restricting the reference picture lists, list0 and list1, for each picture of layer $l$ to temporally preceeding and succeeding pictures, respectively, with a temporal layer identifier less than $l$. In Figure 8.19(a), pictures 0 and 1 (decoding order) belong to $T_0$ (base layer), where picture 1 is undirectionally predicted from 0 (depicted by arrow). Picture 2, predicted from 0 and 1, belongs to $T_1$. Pictures 3 (predicted from 0 and 2) and 6 (predicted from 2 and 1) belong to $T_2$. Pictures 4, 5, 7, and 8, which are predicted from their immediate neighbors, belong to $T_3$. It was found that hierarchical B-pictures not only enable temporal scalability but also provide superior compression efficiency compared to the traditional IBBPBBP... prediction structure when the GOP size is 16 or 32 and the quantization parameter $QP_l$ of layer $l$ is

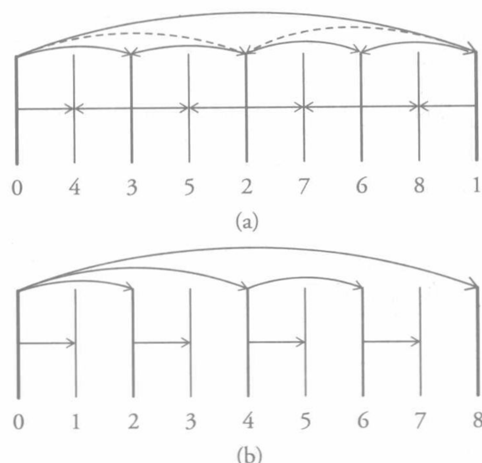



**Figure 8.19** Hierarchical prediction structures: (a) dyadic B-pictures structure and (b) prediction structure for low-delay encoding/decoding.

chosen according to $QP_l = QP_0 + 3 + l$, $l > 0$ [Sch 07]. It is recommended that the "spatial direct" mode of inter-picture prediction be employed when using hierarchical B-pictures with $l > 2$.

The hierarchical B-pictures introduce an encoding/decoding delay of one GOP, which may not be suitable for interactive (videophone) applications. Alternatively, we can use a uni-directional hierarchical prediction structure by using only list0 in hierarchical B-pictures. This causal hierarchical prediction structure, shown in Figure 8.19(b), provides the same level of temporal scalability as the hierarchical B-pictures without introducing a structural delay, but at the expense of reduced compression efficiency.

## 8.5.2 Spatial Scalability

*Spatial (pixel-resolution) scalability* provides the ability to decode video at two or more spatial resolutions without first decoding all the full-resolution frames. The base layer is a low spatial-resolution video. Enhancement layers contain successively higher-resolution video.

Spatial-scalable encoding employs a pyramid representation of each frame. Lower-resolution layers are obtained by successively decimating the full-resolution video. Enhancement layers consist of the difference between the current resolution layer and interpolation of the decoded lower layer video to the size of the current layer. SVC specifies an 11-tap decimation filter given by (1 0 −5 0 20 32 20 0 −5 0 1)/64 and a 6-tap interpolation filter given by (1 −5 20 20 −5 1)/32 for the case

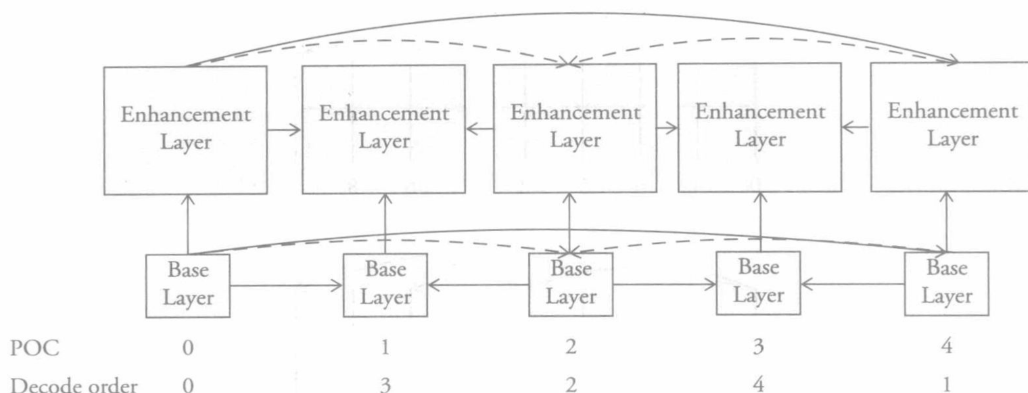



**Figure 8.20** Intra- and inter-layer prediction in spatial-scalable coding.

the resolution of successive layers doubles. The generalized spatial-scalability option of SVC supports arbitrary resolution ratios as well as scalability for interlaced video.

In addition to the single-layer prediction modes in H.264/AVC, SVC specifies new inter-layer prediction schemes where encoding mode, motion vectors, and/or enhancement-layer pixels can be predicted from the decoded lower-resolution layers to improve the coding efficiency (see [Sch 07] for details). Figure 8.20 illustrates inter-layer prediction where enhancement layer pictures can be predicted from interpolated versions of decoded base-layer pictures in addition to hierarchical prediction within each layer. SVC specifies that the same coding order is used in all layers.

In order to limit decoder complexity, SVC introduces mandatory constraints that enable decoding with a single motion-compensation loop. As a result, the complexity overhead introduced by SVC is small compared with those of scalable profiles of previous standards. Spatial-scalable coding may result in a 10% to 50% increase in bit-rate at the same quality, depending on the properties of the specific video and selected prediction structure, when compared to single-layer encoding.

## 8.5.3 Quality (SNR) Scalability

*SNR scalability* offers the ability to decode video with different quality levels, where all layers have the same spatial and temporal resolution. SNR scalability is a valuable tool to achieve error-resilient video streaming over heterogeneous networks, where the base layer may be served using better quality of service parameters and error-correction capabilities, whereas enhancement layers can be served as best-effort service using rate adaptation.

Coarse-grain SNR scalability refers to the case where the number of supported bit-rate points is limited by the number of layers. It can be considered as a special case of spatial scalability where all layers have the same resolution/size without any decimation/interpolation filtering, and inter-layer residual prediction is performed in the transform domain.

Medium-grain SNR scalability (MGS) provides packet-based scalability within a given range of bit-rates in a nearly continuous manner, which is well-suited for network-rate adaptation in video streaming over the Internet, by dropping some of the enhancement layer packets. The base layer is obtained by a coarse quantization of DCT coefficients. Enhancement layers contain DCT-refinement coefficients to progressively increase the quality of decoded video. When decoded video packets (NAL units) can be unpredictably dropped, controlling drift due to encoder-decoder mismatch becomes a key issue. It is possible to completely eliminate drift by disabling the prediction from the enhancement layer (allowing only prediction from the reconstructed base layer), but this approach (used in the MPEG-4 fine granular scalability method) results in a significant loss of compression efficiency. The other extreme is to always allow prediction from the highest-quality enhancement layer (used in the MPEG-2 SNR scalability method), which results in significant drift that can only be managed by frequent intra-frame transmissions. SVC allows encoders to select a suitable tradeoff between compression efficiency and drift control by introducing the notion of key pictures, where the base layer is also stored in the DPB to resynchronize the encoder and decoder reconstruction processes that limits drift to between two key pictures. A combination of hierarchical B-pictures and key pictures (denoted by darker lines) for MGS coding with a GOP size 4 is illustrated in Figure 8.21, where base-layer pictures between two key pictures are predicted from the highest quality (enhancement layer) pictures.

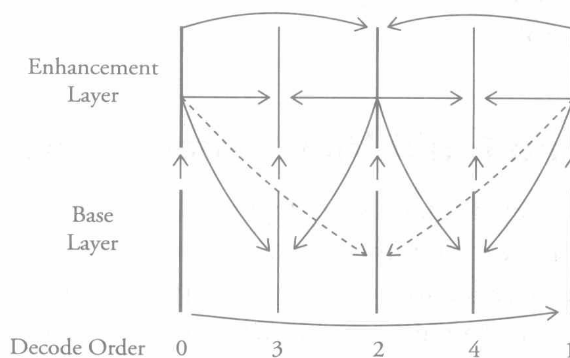



**Figure 8.21** Hierarchical intra- and inter-layer prediction in MGS with key pictures.

Bit-stream extraction refers to obtaining video at a desired bit-rate (lower than the maximum supported rate) by discarding some MGS-encoded NAL units. Clearly, the same desired bit-rate can be attained by discarding different NAL units, which would yield different quality videos. The simplest method is to randomly discard NAL units until the desired bit-rate is reached. A more sophisticated approach is to determine the NAL units to be discarded by using a rate-distortion analysis. SVC syntax supports assigning a priority identifier to each NAL unit by an encoder. Then, NAL units with the lowest priority are discarded first, followed by the next lower priority, and so on. Prioritization of NAL units leads to generation of the so-called priority layers.

Since motion compensation is performed using a single loop, the complexity of a decoder supporting MGS is close to that of a single-layer H.264/AVC decoder. With proper encoder optimization, the bit-rate overhead of MGS encoding is 10% to 20% compared with single-layer H.264/AVC encoding at the same fidelity when the ratio of the highest supported bit-rate to the lowest is between 2 and 3.

### 8.5.4 Hybrid Scalability

*Hybrid scalability* refers to a combination of spatial, temporal, and SNR scalability provided in a single stream. The SVC bit-stream structure is organized in terms of dependency layers, which represent different spatial/temporal resolutions. Quality-refinement layers are defined within each dependency layer to allow for hybrid scalability. When inter-layer prediction is employed, the dependency identifier and the quality identifier of the reference layer must be signaled. Switching between different dependency layers is only allowed at predefined frames, whereas switching between quality layers is possible at any access unit.

SVC also supports region-of-interest (ROI) scalability, which can be implemented using slice groups. However, the shape of the ROI can only be represented as a collection of macroblocks.

## 8.6 Stereo and Multi-View Video Compression

With recent advances in display technologies and video compression, 3D video, which offers immersive entertainment and communication experiences, including free-view TV, has become feasible and highly popular [Tan 12]. 3D-video formats include stereoscopic (stereo-pair), multi-view ($n$-view), and $n$-view plus $n$-depth representations. An overview of 3D-video-compression approaches and standards is provided in [Tan 14]. A summary of these approaches is depicted in Figure 8.22.

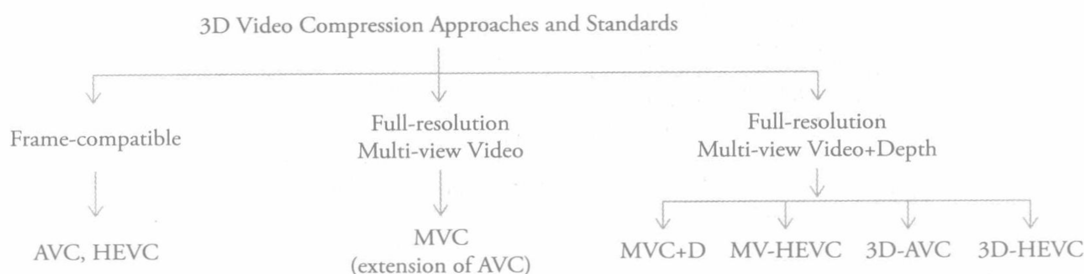



**Figure 8.22** Overview of 3D-video compression.

*Frame-compatible formats* have been developed to compress down-sampled stereo video by using existing (legacy) compression and transmission standards. While this approach is simple, it results in a loss of spatial resolution.

The multi-view video coding (MVC) extension of the H.264/AVC standard was developed for efficient compression of full-resolution stereo *and* multi-view video using inter-view disparity-compensated compression. However, the total bit-rate still increases linearly with the number of views, and more efficient representation and compression schemes are required especially when the number of views is large.

The *n-view plus n-depth format* has proven to be efficient for compression of multi-view video with a large number of views (e.g., 45 or more). Standardization of this format as extensions of the H.264/AVC and HEVC standards is in progress. Besides autostereoscopic free-view 3D-video, multi-view video formats can also be used in free-view 2D video, which can be viewed from multiple angles interactively using a conventional 2D display, and in computational imaging (e.g., synthetic aperture photography).

## 8.6.1 Frame-Compatible Stereo-Video Compression

*Frame-compatible formats*, where the right and left stereo frames are down-sampled and packed together in a single frame, allow adding stereo-video services with only a software upgrade of existing equipment and infrastructure. Some of the common frame-compatible stereo-video formats are illustrated in Figure 8.23, where R and L denote the pixels of the right and left views, respectively. In frame-compatible formats, the spatial resolution of the right and left views is reduced horizontally and/or vertically. Alternatively, it is possible to multiplex right and left views temporally in a *frame/field sequential format*, where the frame/field rates of the right and left views may be reduced by a factor of two.

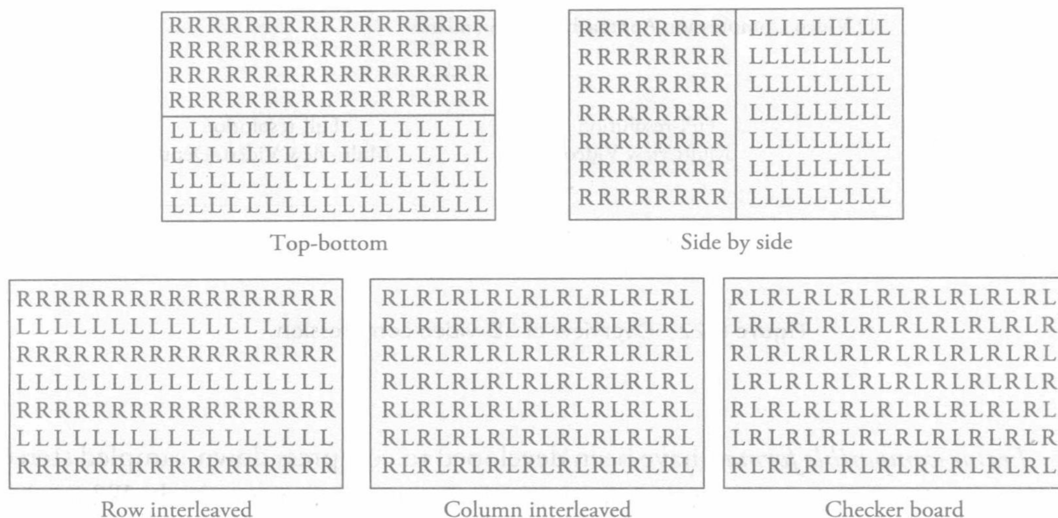



**Figure 8.23** Common frame-compatible stereo-video formats.

Only the display sub-system that follows decoding needs to be modified in the user set-top box to offer 3D-video services to customers with 3D-capable displays over the existing transmission infrastructure using one of these formats. In order to correctly parse the left and right views from a frame-compatible format view, the display sub-system must know the exact packing format. An early version of the H.264/AVC standard (2004) supported stereo-video information (SVI) SEI messages that could signal a row-based interleaving of right-left views vs. a field-sequential ordering of views, as well as whether inter-view prediction is enabled or disabled. The MVC extensions of H.264/AVC and H.265/HEVC now include an extended set of SEI messages, called frame-packing arrangement (FPA) SEI, which can signal all the packing formats depicted in Figure 8.23.

## 8.6.2 Stereo and Multi-View Video-Coding Extensions of the H.264/AVC Standard

The basic approach of stereo and MVC is to exploit redundancies that exist between neighboring views at a given time instant in addition to temporal redundancy between frames of a given view. A recent stereo and MVC extension of the H.264/AVC standard specifies syntax and a set of tools that achieve a significant reduction in bit-rate relative to independent coding of the views without sacrificing the reconstructed video quality.

MVC has been an active research area since the early work on disparity-compensated prediction by Lukacs in 1986 [Luk 86]. The H.262/MPEG-2 standard was amended in 1996 to support MVC by reusing tools originally intended for temporal scalability. However, multi-view extension of MPEG-2 video was never adopted in the market mainly because: i) transition from analog to digital TV and HDTV was a big challenge at that time, ii) 3D-display technology and flat-panel hardware were lacking at that time, and iii) MPEG-2 stereo coding did not offer a compelling compression improvement due to limited coding tools.

A key feature of the MVC extension of H.264/AVC is its high compression efficiency compared with independent coding of views without changing the low-level syntax and decoding process of H.264/AVC. This design constraint enables hardware implementation of MVC decoders with only simple changes to existing H.264/MPEG-4 AVC decoding chipsets. The MVC bit-stream consists of a base view that is coded independently of other views and must conform to one of the H.264/AVC profiles, e.g., High profile, and supplementary views that may be dependent on the base view and each other. The base view is encapsulated in AVC video NAL units, so they can be decoded by legacy AVC decoders. Other views are encapsulated in an extension NAL unit type that is also used for SVC bit-streams. A flag is used to distinguish between SVC and MVC-NAL units, which can be decoded by MVC decoders and discarded by legacy decoders. The main technical novelties in MVC include introduction of *anchor pictures* for efficient inter-view prediction, as well as time-first coding and reference-picture management to achieve low-delay encoding/decoding and optimal memory consumption at the decoder.

MVC introduces a new picture type, called *anchor picture*, which is similar to IDR pictures in that temporal prediction is not allowed; however, inter-view prediction from other views within the same access unit is allowed. Any picture that follows the anchor picture in both decoding order and display order cannot use any picture that precedes the anchor picture in decoding order as a reference for interpicture prediction, and any picture that precedes the anchor picture in decoding order cannot follow it in display order. This provides a clean random-access point for a given view. In MVC, inter-view prediction is adaptive and the best predictor in terms of rate-distortion cost between temporal or inter-view references is chosen on a block basis. It has been observed that most of the coding gain that comes from inter-view prediction is realized at the anchor pictures [Vet 11]. Hence, turning inter-view prediction off at non-anchor frames saves memory and coding delay.

In developing the MVC, two different decoding orders, *view-first coding* and *time-first coding*, have been considered. In view-first coding, pictures of each view

within each group of pictures (GOP) are contiguous in their decoding order. Coded pictures belonging to different views at the same time instant are interleaved with other pictures at other times, and thus cannot be in the same access unit. These different access units, when stored in an ISO base media file, which requires samples to be ordered in their decoding order, have composition time offsets proportional to the GOP size multiplied by the number of views that causes a significant initial buffering delay. Hence, MVC uses *time-first coding*, where pictures at any time instant are contiguous in the bitstream. We can then define pictures at the same time but belonging to different views as one access unit, and an access unit contains NAL units continuous in decoding order [Che 09].

During the development of the MVC, a number of macroblock-level coding tools were also explored, including:

- *Adaptive reference filtering,* which compensates for focus mismatches between different views.
- *View-synthesis prediction,* which predicts a picture in the current view from synthesized references generated from neighboring views to achieve additional coding gains [Yea 09].
- *Illumination compensation,* which compensates for illumination differences as part of the inter-view prediction process.
- *Motion-skip mode,* which infers motion vectors from inter-view references noting the correlation between motion vectors in different views.

While illumination compensation and motion-skip mode offer notable gains, they have not been adopted into the MVC standard because they require changes affecting macroblock level encoding and decoding processes causing implementation concerns [Vet 11].

### Asymmetric Coding of Stereo Video

According to the suppression theory of stereo human vision, the human-vision system can tolerate absence of high frequencies in one of the views; therefore, the left and right views of stereo video can be represented at unequal resolutions or bitrates. Asymmetric coding refers to encoding the non-base view with lower quality than the base view, where the non-base view can be significantly blurred or more coarsely quantized (resulting in bit-rate reduction), or coded with a reduced spatial resolution. Substantial savings in bit-rate can be achieved by using asymmetric coding without a perceptible impact on stereo-video quality. It has been shown that asymmetry by blurring provides finer control over achievable PSNR values [Say 11];

hence, it is superior to asymmetry by spatial-resolution reduction at high bit-rates where it provides better rate-distortion (RD) performance. The MVC standard provides the encoder with the freedom to select the fidelity for each view by performing pre-processing, such as blurring if desired; however, it uses the same sample-array resolution for the encoding of all views. Extensive subjective tests have been conducted to demonstrate the performance of asymmetric coding using short video clips; however, further study is needed to understand whether it would cause eye fatigue in longer duration videos.

## 8.6.3 Multi-View Video Plus Depth Compression

While coding of two fixed views, using a frame-compatible format or MVC as described above, provides basic 3D perception on stereoscopic displays, it is not suitable for disparity adjustment between views for adaptation to different displays and viewing conditions, which has been shown to provide a superior 3D experience. Moreover, state-of-the-art auto-stereoscopic displays require displaying a large number of views (e.g., 45 or more), which would require significant bit-rates if all views were encoded by MVC. The *multi-view video plus depth* (MVD) format is a versatile 3D-video format that consists of a small number of views (texture) and their associated depth maps [Mul 11]. The depth value at each pixel is represented by monocular-depth images that have minimal high-frequency content and can be compressed efficiently. Using the MVD, stereo-disparity adjustment or additional intermediate view synthesis can be computed at the decoder using depth-image-based rendering (DIBR). We can classify recent work on MVD compression into two groups: i) those compatible with the H.264/AVC standard [Han 13] and (ii) those compatible with the H.265/HEVC standard [Mul 13]. The former group includes MVC+D and 3D-AVC configurations, which are explained below.

#### Multi-View Coding Extensions for Inclusion of Depth Maps

MVC+D, finalized in January 2013, is a backward-compatible extension of the MVC standard for inclusion of depth maps. It specifies the encapsulation of MVC-coded texture views and depth maps into a single bit-stream [Han 13]. Texture only views of MVC+D bit-streams can be decoded with an MVC decoder. The depth maps, together with the high-level syntax signaling the necessary information for interpretation of the depth data, are represented by an independent second stream, which can be encoded by MVC as if it were a multi-view monochrome video. Interlace coding for texture or depth is supported for stereo views.

MV-HEVC is an extension of the HEVC standard to provide efficient representation of multi-view video and optional depth map information, e.g., for 3D stereoscopic and autostereoscopic video applications. It was finalized in July 2014.

### 3D-AVC

3D-AVC is an advanced coding process that jointly encodes dependent (non-base) views and depth images to achieve better compression efficiency than MVC+D. 3D-AVC encodes a base view that is backward compatible with the H.264/AVC standard and independent of other views to support legacy monoscopic receivers. We note that a decoder supporting 3D-AVC can also decode MVC+D bit-streams.

A 3D-AVC access unit is formed by all the video-texture and depth-map-view components that describe a 3D scene at a particular time instant. The data of a coded-view component is not interleaved by any other coded-view component, and the data for an access unit is not interleaved by any other access unit in the bit-stream/decoding order. The AVC/MVC-compatible video texture view components are coded before the respective depth-view components. Enhanced video-texture-view components are coded after the respective depth-view components. The video-texture and depth-view components of the same access units are coded in view-dependency order. Examples of coding order for an access unit include:

1. T0, D0, T1, D1, ... (two AVC/MVC-compatible texture views)
2. T0, D0, D1, T1, ... (an AVC-compatible view, an enhanced texture view)

where T and D denote texture and depth map and the numerals indicate view numbers.

The 3D-AVC specification includes several advanced coding tools that are briefly described in the following. These advanced coding tools do not support interlaced-video coding [Han 13].

1. Baseline-depth coding tools:
   a. Non-linear depth representation (NDR): This tool enables representing closer objects more accurately than distant ones. If NDR is turned on, the depth map is nonlinearly mapped through a forward lookup table at the preprocessing stage of the encoder and inversely mapped back to the original representation at the post-processing stage of the decoder. This tool is none-normative for MVC+D.
   b. Reduced-resolution depth coding: Flexible depth-to-texture resolution ratio is allowed, e.g., depth resolution equal to 1/2 of luma resolution vertically

and horizontally. Encoder can control the depth resolution relative to the luma-texture resolution with horizontal and vertical scale factors as well as shift values in the 3D-AVC sequence parameter set.

2. Enhanced-depth coding tools:
   a. Depth-Range-Based Weighted Prediction (DRWP): This tool performs a non-linear compensation of the depth map.
   b. In-Loop Joint Inter-View Depth Filtering (JVDF): Depth-map images of available views are filtered jointly.
   c. Motion prediction from texture to depth: Since a texture view and its associated depth-view component have similar objects, there is redundancy in their motion fields. This tool can be applied only for depth views for AVC/MVC-compatible texture views.
   d. Depth Intra-Prediction includes depth intra-skip prediction and plane-segmentation-based intra-prediction (PSIP).
3. Enhanced texture-coding tools applicable to dependent video views:
   a. In-Loop Block-Based View Synthesis Prediction (VSP): A decoded texture-view component is projected to the viewing point of the current (de)coded dependent view using DIBR, given the camera parameters. The projected image is included in the reference picture list(s) and serves as a reference for MCP.
   b. Depth-Based Motion Vector Prediction (DMVP): This tool consists of direction-separated motion-vector prediction for the inter-mode and disparity-based skip and direct modes for further improving the accuracy of motion-vector predictors.
   c. Inter-view coding with adaptive luminance compensation (ALC): This tool suppresses local illumination changes between encoded macroblocks and predicted blocks that belong to an interview reference frame.

   VSP and DMVP perform joint texture and depth coding, where samples of depth data are utilized for efficient coding of texture. This introduces an inter-component dependency.
4. "Slice header prediction" can be used for both enhanced-depth and enhanced-texture views.
5. Non-normative tools that can be used with MVC+D and 3D-AVC:
   a. Gradual view refresh (GVR) for texture and depth coding
   b. Rate-distortion optimization through view synthesis distortion (VSD)
   c. Post-processing dilation filtering (PDF) for depth map

## 3D-HEVC

The 3D-HEVC encoder takes multiple views, associated depth maps, and corresponding camera parameters as input, although it can also operate without depth data. Various scenarios for the use of 3D-HEVC codec are depicted in Figure 8.24. At the decoder, additional intermediate views can be rendered by a view synthesizer for display on a multi-view auto-stereoscopic display. View synthesis can be performed by a DIBR algorithm using the reconstructed views and depth data or by image-domain warping without depth data. The encoder can be configured such that a sub-bit-stream containing only two stereo views can be extracted and decoded using a stereo decoder. The view synthesizer can also render a stereo pair for a conventional stereo display, in case a stereo pair is not present in the bit-stream or to adjust the stereo views to the viewing conditions. The base view, which can be extracted and decoded by using an unmodified HEVC decoder, or one of the views decoded by 3D-HEVC decoder or a synthesized intermediate view at an arbitrary virtual camera position, can also be displayed on a conventional 2D display.

A 3D-HEVC access unit includes all video pictures and depth maps that correspond to the same time instant (time-first coding). NAL units containing camera parameters may also be associated with an access unit. The reconstructed data of already-coded access units can be used for coding the current access unit. Random access is enabled by IDR access units.

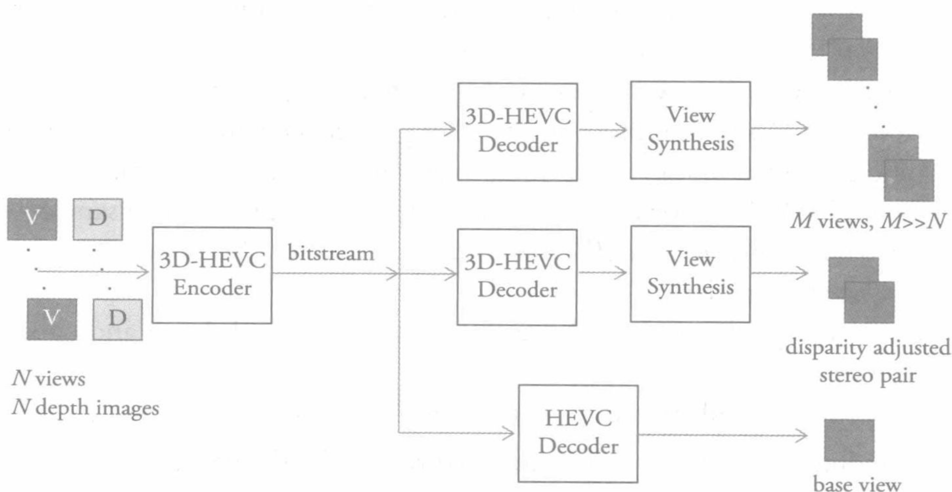



**Figure 8.24** 3D-HEVC use scenarios.

The 3D-HEVC offers the following new tools:

1. Video View Coding: New tools for coding dependent views include disparity-compensated prediction (DCP), advanced residual prediction (ARP), illumination compensation (IC), view-synthesis prediction (VSP), and depth-based block partitioning (DBBP). The well-known concept of DCP, also used in MVC and 3D-AVC, was added as an alternative to MCP. ARP is a coding tool to exploit the residual correlation between views. IC uses a linear model to adapt luminance and chrominance of inter-view predicted blocks to the illumination of the current view. VSP provides a predictor using depth information to reduce inter-view redundancy. DBBP partitions the collocated texture block based on a binary segmentation mask computed from the depth map. Each of the two partitions (e.g., foreground and background) is motion compensated and then merged using a depth-based segmentation mask.
2. Depth-Map Coding: Depth-map coding employs the same intra-prediction, MCP, disparity-compensated prediction, and transform coding concepts as video coding. However, some tools have been modified for depth maps, other tools have been generally disabled, and additional tools have been added. The inter-view motion, residual prediction, view synthesis prediction, and in-loop filters are not used for depth coding. Instead, motion parameters are derived based on coded data in the associated video pictures. New additions include new intra-coding modes, modified motion compensation and motion-vector coding, and motion-parameter inheritance.
3. Motion Coding: Inter-view motion prediction derives motion parameters for a block in a current picture based on motion parameters in an already coded reference view and an estimate of the depth map for the current picture. The motion data is compressed into 1/4 resolution after encoding/decoding of each picture and then further compressed into 1/16 resolution after encoding/decoding of all the pictures within the same access unit in order to reduce buffer size and memory bandwidth.
4. Encoder Control: For mode decision and motion estimation, similar to encoder control in other standards, a Lagrangian technique that minimizes a cost measure $D + \lambda \cdot R$, where $D$ is the decoding distortion of a particular mode with particular parameters for the considered block, $R$ is the number of bits required for coding the block with these parameters, and $\lambda$ is the Lagrangian multiplier that is a function of the quantization parameter, is used for each candidate mode or parameters, and the mode or parameters with the smallest cost

measure is selected. As measure of the distortion, the sum of squared differences (SSD) or the sum of absolute differences (SAD) between the original and the reconstructed sample values is used. For the coding of depth maps, the same decision process is used. However, the distortion measure was replaced with a measure that considers the distortion in synthesized intermediate views using an encoder-side render model.

5. Decoder-Side View Synthesis: A fast 1D-view synthesis method based on DIBR for generating the required number of display views is provided.

## References

[Aiz 95] Aizawa, K., and T.S. Huang, "Model-based image coding: Advanced video coding techniques for very low bit-rate applications," Proc. of the IEEE, vol. 83, no. 2, pp. 259–271, Feb. 1995.

[Bos 92] Bosveld, F., R. L. Lagendijk, and J. Biemond, "Compatible spatio-temporal subband encoding of HDTV," Signal Proc., vol. 28, pp. 271–290, Sep. 1992.

[Che 93] Chen, C.-T., "Video compression: Standards and applications," Visual Comm. Image Rep., vol. 4, no. 2, pp. 103–111, June 1993.

[Che 09] Chen, Y., Y.-K. Wang, K. Ugur, M.M. Hannuksela, J. Lainema, and M. Gabbouj, "The emerging MVC standard for 3D video services," EURASIP J. on Advances in Signal Processing, Hindawi, vol. 2009, 13 pages, 2009.

[Chi 95] Chiariglione, L., "The development of an integrated audiovisual coding standard: MPEG," Proc. of the IEEE, vol. 83, no. 2, pp. 151–157, Feb. 1995.

[Cho 99] Choi, S., and J.W. Woods, "Motion compensated 3D subband coding of video," IEEE Trans. on Image Proc. vol. 8, pp. 155–167, 1999.

[Gal 91] LeGall, D. J., "MPEG: A video compression standard for multimedia applications," Commun. of the ACM, vol. 34, pp. 46–58, 1991.

[Gal 92] LeGall, D. J., "The MPEG video compression algorithm," Signal Proc.: Image Comm., vol. 4, pp. 129–140, 1992.

[Ham 14] Hamidouche, W., M. Raulet, and O. Deforges, "Parallel SHVC decoder: Implementation and analysis," Proc. IEEE Int. Conf. Multimedia and Expo (ICME), pp. 1–4, Chengdu, China, July 2014.

[Han 13] Hannuksela, M. M., D. Rusanovskyy, W. Su, L. Chen, R. Li, P. Aflaki, D. Lan, M. Joachimiak, H. Li, and M. Gabbouj, "Multiview-video-plus-depth coding based on the advanced video coding standard," IEEE Trans. on Image Proc., vol. 22, no. 9, Sept. 2013.

[Has 72] Haskell, B. G., F. W. Mounts, and J. C. Candy, "Interframe coding of videotelephone pictures," Proc. of the IEEE, vol. 60, pp. 792–800, July 1972.

[ISO 13] ISO/IEC 13818-2:2013, Information technology—Generic coding of moving pictures and associated audio information—Part 2: Video, 2013.

[Luk 86] Lukacs, M. E., "Predictive coding of multi-viewpoint image sets," Proc. IEEE ICASSP, vol. 1, pp. 521–524, Tokyo, Japan, 1986.

[Luo 94] Luo, J., C. W. Chen, K. J. Parker, and T. S. Huang, "Three dimensional subband video analysis and synthesis with adaptive clustering in high-frequency subbands," Proc. IEEE Int. Conf. Image Processing, Austin, TX, Nov. 1994.

[Mar 03] Marpe, D., V. George, H.L. Cycon, and K.U Barthel, "Performance evaluation of motion JPEG 2000 in comparison with H264/AVC operated in pure intra coding mode," Proc. SPIE Int. Symp. Photonics Tech. for Robotics, Automation and Manufacturing: Wavelet Apps. in Industrial Processing, vol. SPIE 5266, Oct. 2003.

[Mis 13] Misra, K., A. Segall, M. Horowitz, S. Xu, A. Fuldseth, and M. Zhou, "An overview of tiles in HEVC," IEEE J. of Selected Topics in Signal Processing, vol. 7, no. 6, pp. 969–977, Dec. 2013.

[Mul 11] Müller, K., P. Merkle, and T. Wiegand, "3D video representation using depth maps," Proceedings of the IEEE, vol. 99, no. 4, pp. 643–656, April 2011.

[Mul 13] Müller, K., H. Schwarz, D. Marpe, C. Bartnik, S. Bosse, H. Brust, T. Hinz, H. Lakshman, P. Merkle, F.H. Rhee, G. Tech, M. Winken, and T. Wiegand, "3D high-efficiency video coding for multi-view video and depth data," IEEE Trans. on Image Proc., vol. 22, no. 9, pp. 3366–3378, Sept. 2013

[Ohm 12] Ohm, J.R., G.J. Sullivan, H. Schwarz, T.K. Tan, and T. Wiegand, "Comparison of the coding efficiency of video coding standards including High Efficiency Video Coding (HEVC)," IEEE Trans. Circuits and Systems for Video Tech., vol. 22, no. 12, pp. 1669–1684, Dec. 2012.

[Pen 93] Pennebaker, W. B., and J. L. Mitchell, *JPEG: Still Image Data Compression Standard*, New York, NY: Van Nostrand Reinhold, 1993.

[Roe 77] Roese, J. A., W. K. Pratt, and G. S. Robinson, "Interframe cosine transform image coding," IEEE Trans. Comm., vol. 25, no. 11, pp. 1329–1339, Nov. 1977.

[Say 11] Saygili, G., C. G. Gurler, and A. M. Tekalp, "Evaluation of asymmetric stereo video coding and rate scaling for adaptive 3D video streaming," IEEE Trans. on Broadcasting, vol. 57, no. 2, pp. 593–601, June 2011.

[Sch 07] Schwarz, H., D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the H.264/AVC standard," IEEE Trans. on Circuits and Systems for Video Tech., vol. 17, no. 9, pp. 1103–1120, Sep. 2007.

[Sul 05] Sullivan, G.J., and T. Wiegand, "Video compression - From concepts to the H.264/AVC standard," *Proc. of the IEEE,* vol. 93, no. 1, pp. 18–31, Jan. 2005.

[Sul 12] Sullivan, G.J., J.R. Ohm, W.J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," IEEE Trans. on Circuits and Systems for Video Tech., vol. 22, no. 12, pp. 1649–1668, Dec. 2012.

[Tan 12] Tanimoto, M., "FTV: Free-viewpoint television," Signal Proc.: Image Comm., vol. 27, pp. 555–570, 2012.

[Tan 14] Tanimoto, M., "FTV standardization in MPEG," Proc. IEEE 3DTV-Conference, pp. 1–4, Budapest, Hungary, July 2014.

[Vet 92] Vetterli, M., and K. M. Uz, "Multiresolution coding techniques for digital television: A review," Multidim. Syst. Signal Proc., vol. 3, pp. 161–187, 1992.

[Vet 11] Vetro, A., T. Wiegand, and G.J. Sullivan, "Overview of the stereo and multi-view video coding extensions of the H.264/MPEG-4 AVC standard," Proceedings of the IEEE, vol. 99, no. 4, pp. 626–642, April 2011.

[Wie 03] Wiegand, T., G.J. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," IEEE Trans. on Circuits and Systems for Video Tech., vol. 13, no. 7, pp. 560–576, July 2003.

[Yea 09] Yea, S., and A. Vetro, "View synthesis prediction for multiview video coding," Signal Processing: Image Communication, vol. 24, nos. 1–2, pp. 89–100, Jan. 2009.

## Exercises

**8.1** The types of frames in a group of pictures (GOP) of MPEG-2 coded video and their display orders are shown below. Write the coding/decoding order for this GOP, and show the reference frames used in motion compensating each frame.

| I | B | B | B | B | B | B | B | B | B | B | B | B | B | B | B |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

**8.2** Explain intra-prediction in H.264/MPEG-4 AVC. Discuss different options briefly.

**8.3** Show the coding/decoding order for the following GOP in H264/AVC video coding using hierarchical B-pictures, and show the reference frames used for motion compensating each frame.

| I | B | B | B | B | B | B | B | B | B | B | B | B | B | B | B |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

8.4 Compare motion JPEG2000 vs. all-intra-AVC video encoding conceptually. Which one do you think would have higher compression efficiency?

8.5 What tools can be used in an AVC encoder to avoid error propagation in transmission of coded video over error-prone networks? Explain in detail.

8.6 What is rate control in video coding? When do we need rate control in video encoding? Describe a simple rate-control algorithm.

8.7 Discuss constant-quality vs. constant-rate video encoding. How do you achieve constant-quality video encoding? Can you achieve constant-quality encoding while satisfying a desired average bit-rate?

8.8 What is drift in scalable video coding? What is a key picture? Discuss approaches to avoid drift with their effect on coding efficiency.

8.9 Discuss the overhead bit-rate requirements for temporal, spatial, and quality SVC. Discuss degradation introduced by dropping enhancement layers in each type of scalability for different types of video content.

8.10 Discuss the pros and cons of SVC vs. adaptive stream (bit-rate) switching for video streaming over the Internet.

8.11 Discuss the bit-rate requirements for coding multi-view video as a function of distance between cameras. How does the bit-rate requirement vary by the number of views?

8.12 Discuss the bit-rate requirements for coding depth-map images compared to video views. Explain why is it easier or more difficult to encode depth maps.

## Internet Resources

VideoLAN x.264 Free Software Library and Application
http://www.videolan.org/developers/x264.html

x.265 Open Source Project
http://x265.org/

Web-M project, VP8 and VP9 codecs
http://www.webmproject.org/

APPENDIX A

# Ill-Posed Problems in Image and Video Processing

Many image/video-processing problems are ill-posed whose analysis and solution requires a strong mathematical foundation and proper image/video/motion modeling. This appendix aims to summarize fundamental modeling approaches in modern image and video processing, and provide the foundation for sparse image representations, to help put some well-known regularization methods for solving ill-posed image- and video-processing problems on a common framework.

## A.1 Image Representations

### A.1.1 Deterministic Framework – Function/Vector Spaces

We can represent a discrete image $s(n_1, n_2)$ by an $N \times N$ matrix of pixels or by an $N^2 \times 1$ vector $\mathbf{s} = [s_1\ s_2\ \ldots\ s_{N^2}]^T$ by mapping pixels $(n_1, n_2)$ into a 1D order $j = 1, \ldots, N^2$, e.g., by lexicographic ordering. For video, we add a time dimension with causal ordering in time. We assume the inner product and norm are defined in this (Hilbert) space in the usual sense. Then, a specific image/video processing problem can be formulated as optimization of $l^2$-, $l^1$-, or $l^0$-norm of a suitable error

function subject to some constraints. These norms can also be used to induce some probability distribution over an ensemble of images as will be discussed next.

### A.1.2 Bayesian Framework – Random Fields

Let's now consider a family or ensemble of images, such as all natural images or all smooth images, $\mathcal{S} = \{\mathbf{s}_i, i = 0, 1, 2 \ldots\}$, such that $\mathbf{s}_i \in R^{N^2}$. If we assume that pixels $j$ have values $s_j$ in the range [0, 1), images in an ensemble do not populate the hyper-cube $[0, 1)^{N^2} \subset R^{N^2}$ uniformly. We can model the distribution of images in a particular ensemble by using a probability density function (pdf), called the *a priori* distribution $P(\mathbf{s})$, which leads to the Bayesian framework for image processing, where we typically optimize the mean-square error (minimum mean-square error estimation), the conditional probability distribution (maximum-likelihood estimation), or *a posteriori* probability distribution (maximum *a posteriori* probability estimation) subject to some constraints. The extension of the Bayesian framework for video is straightforward.

## A.2 Overview of Image Models

Many image- and video-processing problems are ill-posed in the sense that the solution is not unique, and/or it is highly sensitive to the presence of noise. Examples of such problems include image-gradient estimation; interpolation; inverse problems (including de-noising, restoration, super-resolution, and inpainting) of the form $\mathbf{y} = \mathbf{H}\mathbf{s} + \mathbf{v}$, where $\mathbf{H}$ is an over-determined or under-determined matrix and $\mathbf{v}$ is observation noise; and 2D/3D motion estimation/tracking. In general, it is not possible to find an acceptable solution $\mathbf{s}$ to these problems without making some assumptions about the nature of the solution, i.e., employing *a priori* models of the solution, which is called regularization. A regularization method transforms an ill-posed problem to a well-posed problem, whose solution is an acceptable approximation to the solution of the ill-posed problem. For example, for image interpolation, we assume the image is bandlimited; for gradient estimation, we assume the image is smooth; for inverse problems, we assume the image is smooth and/or sparse in some transform domain; for motion estimation/tracking, we assume the motion is continuous over time, etc. The goodness of the solution depends on the suitability of the model used to solve the problem at hand. We can broadly classify image/video/motion models as:

1. Smoothness Models: Perhaps the simplest model used in the image-processing and computer-vision community is the assumption that the solution varies

slowly over space, i.e., it is smooth. In a deterministic framework, smoothness can be modeled by minimizing $||\mathbf{L}\ \mathbf{s}||_2^2$, where $||\cdot||_2$ denotes $l^2$-norm and $\mathbf{L}$ is a Laplacian matrix (defines a linear space-invariant filter applied to the image $\mathbf{s}$) subject to some observation constraint. In a probabilistic framework, this can be expressed as a homogeneous random field with Gibbs *a priori* probability distribution, given by $P(\mathbf{s}) \approx \exp\{-\lambda||\mathbf{Ls}||_2^2\}$. Then, deviation from spatial smoothness, measured by the Laplacian operator, is used as a measure of likeliness of a solution. Note that the smoother the image, the smaller $||\mathbf{Ls}||_2^2$; hence, smoother images are more likely. This prior is well-known to be related to both Tikhonov regularization and Wiener filtering and is extensively used in image processing. We also note that Gibbs random fields are a particular instance of the more general class of image models known as Markov random felds (see Appendix B).

2. Edge (Singularity)-Preserving Models: The smoothness prior defined in terms of the Laplacian and $l^2$-norm is known to cause image over-smoothing when used in image denoising, inverse problems, and motion estimation. The $l^2$-norm strongly penalizes (i.e., makes highly unlikely) any large local differences such as edges and motion discontinuities, which are key features for visual perception. A possible remedy is replacement of the $l^2$-norm by a more robust measure such as the $l^1$-norm that penalizes large values less, and the resulting pdf is allowed to have heavy tails. Thus, a prior of the form $P(\mathbf{s}) \approx \exp\{-\lambda||\mathbf{L}\ \mathbf{s}||_1\}$ has recently become popular. Alternatively, the "total-variation" prior also promotes smoothness by replacing the Laplacian with gradient norms, thereby using first derivatives rather than the second.

3. Sparse Models: Sparse and low-rank representations have recently become very popular in image processing. The sparsity of an image representation can be measured by the $l^0$-norm of the coefficients of a representation using a complete or over-complete dictionary, which corresponds to the number of non-zero components in this representation. More formally, $||\mathbf{T}\ \mathbf{s}||_0 = \lim_{p\to 0}||\mathbf{T}\ \mathbf{s}||_p^p = \{\#\ i, (\mathbf{Ts})_i \neq 0\}$. Optimization of $l^0$-norm results in an NP-complete problem that is intractable. Interestingly, a convex relaxation of such problems can be formulated in terms of the $l^1$-norm, which also enforces sparsity. That is, we can minimize $||\mathbf{Ts}||_1$ subject to some observation constraint (data term) to promote a sparse solution. Note that sparsity of the orthogonal-wavelet transform has been well exploited for image denoising using wavelet shrinkage. In the Bayesian framework, the orthogonal wavelet transform of an image $\mathbf{s}$, given by $\mathbf{Ts}$, can be used to define an image prior $P(\mathbf{s}) \approx \exp\{-\lambda||\mathbf{Ts}||_p^p\}$, with $p \leq 1$ to promote sparsity.

## A.3 Basics of Sparse-Image Modeling

There has been much work on transform domain representation of images including 2D-DFT, 2D-DCT, and 2D-wavelet transforms, where transform-domain representations consist of a set of expansion coefficients with respect to some basis images. Image representations in the transform domain are usually sparse, meaning images can be represented by a small number of coefficients, which is the basic principle of image-compression methods. Recently, such transform-domain representations have been extended to redundant (overcomplete) image representations using a dictionary $\mathbf{D}$ of atomic images, where an $N^2 \times 1$ image $\mathbf{s}$ can be expressed as

$$\mathbf{s} = \mathbf{D}\boldsymbol{\alpha} + \mathbf{e}$$

The dictionary $\mathbf{D}$ is an $N^2 \times M$ matrix, where $M \geq N^2$ (it is redundant for $M > N^2$) and each column represents an atomic image. The vector $\mathbf{e}$ denotes the model error (mismatch) with finite energy, i.e., $||\mathbf{e}||_2^2 < \delta^2$. The vector $\boldsymbol{\alpha}$ is called a sparse representation (most of its entries are zero) based on this dictionary. The sparsity of $\boldsymbol{\alpha}$ is measured by its $l^0$-norm $||\boldsymbol{\alpha}||_0$.

How do we find the sparsest vector $\boldsymbol{\alpha}$ that models $\mathbf{s}$ as a linear combination of columns from $\mathbf{D}$ with an error energy no larger than $\delta^2$? An exact optimal solution of the problem

$$\hat{\alpha} = \arg\min || \alpha ||_0 \text{ subject to } || \mathbf{s} - \mathbf{D}\alpha ||_2^2 \leq \delta^2 \tag{A.1}$$

is computationally intractable. There are two approaches to reach an acceptable solution: i) employ greedy methods, such as the basis pursuit [Che 01], to find a suboptimal solution, and ii) solve a convex relaxation of the problem (A.1) in terms of the $l^1$-norm,

$$\hat{\alpha} = \arg\min || \alpha ||_1 \text{ subject to } ||\mathbf{s} - \mathbf{D}\alpha ||_2^2 \leq \delta^2 \tag{A.2}$$

which also enforces sparsity.

This basic model should often be tuned to specific images/applications by applying it locally and selecting the best parameters that describe the local image characteristics and by choosing an appropriate dictionary, which can be universal or learned from a set of example images.

# A.4 Well-Posed Formulations of Ill-Posed Problems

Well-posed formulations of ill-posed problems can be obtained by regularization approaches. Most ill-posed image- and video-processing problems can be regularized by formulating them as constrained optimization (deterministic framework) or Bayesian estimation (stochastic framework) problems. As an example, let's take inverse problems that can be formulated as: given $\mathbf{y} = \mathbf{Hs} + \mathbf{v}$, where the degradation matrix $\mathbf{H}$ is known and observation noise $\mathbf{v}$ has finite energy, i.e., $\|\mathbf{v}\|_2^2 \leq \sigma^2$, estimate $\mathbf{s}$.

## A.4.1 Constrained-Optimization Problem

The classical constrained least-squares estimation formulation seeks the smoothest image that satisfies the observation equation constraint

$$\hat{\mathbf{s}} = \arg\min_{\mathbf{s}} \|\mathbf{L}\ \mathbf{s}\|_2^2 \ \text{ subject to } \|\mathbf{y} - \mathbf{Hs}\|_2^2 \leq \sigma^2 \tag{A.3}$$

The constraint expresses prior knowledge on the degradation and noise, which limits the solution space. We can employ the Lagrangian method to convert a constrained optimization problem into an unconstrained optimization problem by defining the constraint (data term) as a penalty term

$$\hat{\mathbf{s}} = \arg\min_{\mathbf{s}} \|\mathbf{L}\,\mathbf{s}\|_p^p + \lambda\left(\|\mathbf{y} - \mathbf{Hs}\|_2^2 - \sigma^2\right) \tag{A.4}$$

for $p=1,2$. This unconstrained-optimization problem can either be solved analytically by differentiating the cost function with respect to unknowns and setting the resulting equations equal to zero, or numerically by one of the methods discussed in Appendix C.

Alternatively, the problem has been formulated recently using sparse-image representations as

$$\hat{\mathbf{s}} = \arg\min \|\boldsymbol{\alpha}\|_p^p \ \text{ subject to } \|\mathbf{y} - \mathbf{HD}\boldsymbol{\alpha}\|_2^2 \leq \sigma^2 \tag{A.5}$$

for $p=0,1$, emphasizing sparseness of the solution in some transform domain. The solution of inverse problems using sparse models has been discussed in [Fig 03, Dau 04, Com 05, Ela 10].

## A.4.2 Bayesian-Estimation Problem

Bayesian-estimation problems are commonly stated as maximum-likelihood (ML), maximum *a posteriori* (MAP), or minimum mean-square error (MMSE) estimation problems. Most image- or motion-estimation problems can be posed as a MAP estimation problem defined as

$$\hat{\mathbf{s}} = \arg\max P(\mathbf{s} \mid \mathbf{y}) = \arg\max P(\mathbf{y} \mid \mathbf{s}) P(\mathbf{s}) \tag{A.6}$$

which seeks for the most probable image $\hat{\mathbf{s}}$ in the sphere $\|\mathbf{y} - \mathbf{Hs}\|_2^2 \leq \sigma^2$. The pdf $P(\mathbf{s}|\mathbf{y})$ denotes the *a posteriori* probability distribution of the unknown $\mathbf{s}$ (image or motion vectors) given the observations $\mathbf{y} = \mathbf{Hs} + \mathbf{v}$. The pdf $P(\mathbf{y}|\mathbf{s})$ is called the conditional probability distribution of the observations given $\mathbf{s}$; hence, it models the distribution of the observation noise and enforces the data term, i.e., the constraint $\|\mathbf{y} - \mathbf{Hs}\|_2^2 \leq \sigma^2$. The pdf $P(\mathbf{s})$ is the *a priori* signal model that is discussed in Section A.2. Te optimization of (A.6) is often implemented by one of the numerical optimization schemes discussed in Appendix C.

# References

[Che 01] Chen, S., D. Donoho, and M. Saunders, "Atomic decompositions by basis pursuit," SIAM Review, vol. 43, pp. 129–159, 2001.

[Com 05] Combettes, P., and V. Wajs, "Signal recovery by proximal forward-backward splitting," SIAM Journal on Multiscale Modeling & Simulation, vol. 4, pp. 1168–1200, 2005.

[Dau 04] Daubechies, I., M. Defriese, and C. De Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," Communications on Pure and Applied Mathematics, vol. LVII, pp. 1413–1457, 2004.

[Ela 10] Elad, M., M.A.T. Figueiredo, and Y. Ma, "On the role of sparse and redundant representations in image processing," Proc. of the IEEE, vol. 98, no. 6, pp. 972–982, June 2010.

[Fig 03] Figueiredo, M., and R. Nowak, "An EM algorithm for wavelet-based image restoration," IEEE Trans. on Image Processing, vol. 12, no. 8, pp. 906–916, 2003.

APPENDIX B

# Markov and Gibbs Random Fields

Markov random fields (MRFs) specified in terms of Gibbs distributions have become popular as *a priori* models in Bayesian formulations for image-processing applications such as texture modeling and generation [Cro 83, Che 93], image segmentation and restoration [Gem 84, Der 87, Pap 92], and motion estimation [Dub 93]. This appendix provides the definitions of an MRF and the Gibbs distribution and then describes their relationship by using the Hammersley–Clifford theorem. The specification of MRFs in terms of Gibbs distributions has led to the term "Gibbs random field" (GRF). We also discuss how to obtain the local (Markov) conditional pdfs from the Gibbs distribution, which is a joint pdf.

## B.1 Equivalence of Markov Random Fields and Gibbs Random Fields

We start with defining a random field. A scalar *random field* $\mathbf{z} = \{z(\mathbf{x}), \mathbf{x} \in \Lambda\}$ is a stochastic process defined over a lattice $\Lambda$. Let $\Omega$ denote a realization of the random field $\mathbf{z}$. Recall that the random field $z(\mathbf{x})$ evaluated at a fixed location $\mathbf{x}$ is a random variable. It follows that a scalar random field is a collection of scalar random

variables, where a random variable is associated with each site of the lattice $\Lambda$. A vector random field, such as velocity or displacement fields, is likewise a collection of random vectors. In the following, we limit our discussion to scalar random fields. A random field $\mathbf{z}$ can be discrete-valued or real-valued. For a discrete-valued random field, $z(\mathbf{x})$ assumes a set of discrete values, i.e., $z(\mathbf{x}) \in \Gamma = \{0, 1, \ldots, L-1\}$, and for a real-valued random field $z(\mathbf{x}) \in \mathbf{R}$, where $\mathbf{R}$ denotes the set of real numbers.

The first step in defining MRFs and Gibbs distributions is to develop a *neighborhood system* on $\Lambda$. Let $N_\mathbf{x}$ denote a neighborhood of a site $\mathbf{x} \in \Lambda$ with the properties:

1. $\mathbf{x} \notin N_\mathbf{x}$ and
2. $\mathbf{x}_i \in N_{\mathbf{x}_j} \leftrightarrow \mathbf{x}_j \in N_{\mathbf{x}_i}$, for all $\mathbf{x}_i, \mathbf{x}_j \in \Lambda$

In words, a site $\mathbf{x}$ does not belong to its own set of neighbors, and if $\mathbf{x}_i$ is a neighbor of $\mathbf{x}_j$, then $\mathbf{x}_j$ must be a neighbor of $\mathbf{x}_i$. A neighborhood system $N$ over $\Lambda$ is defined as $N = \{N_\mathbf{x}, \mathbf{x} \in \Lambda\}$, the collection of neighborhoods of all sites. Two examples of neighborhood $N_\mathbf{x}$ of a site are depicted in Figure B.1.

## B.1.1 Markov Random Fields

MRFs are extensions of 1D causal Markov chains to 2D, and have been found useful in image modeling and processing. MRFs have been traditionally specified in terms of local conditional probability density functions (pdfs), which limits their utility.

**Definition.** The random field $\mathbf{z} = \{z(\mathbf{x}), \mathbf{x} \in \Lambda\}$ is called an MRF with respect to $N$ if

$$p(\mathbf{z}) > 0 \text{ for all } \mathbf{z}$$

and

$$p\big(z(\mathbf{x}_i) \mid z(\mathbf{x}_j), \text{all } \mathbf{x}_j \neq \mathbf{x}_i\big) = p\big(z(\mathbf{x}_i) \mid z(\mathbf{x}_j), \text{only } \mathbf{x}_j \in N_{\mathbf{x}_i}\big)$$

where the pdf of a discrete-valued random variable/field is defined in terms of a Dirac delta function.

The first condition states that all possible realizations should have non-zero probability, while the second requires that the local conditional pdf at a site $\mathbf{x}_i$ depends only on the values of the random field within the neighborhood $N_{\mathbf{x}_i}$ of that site.

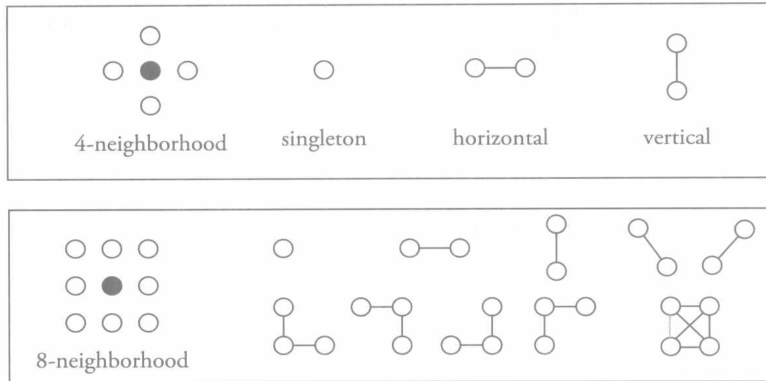



**Figure B.1** Examples of neighborhoods: (a) 4-pixel neighborhood and associated cliques and (b) 8-pixel neighborhood and associated cliques.

The specification of an MRF in terms of local conditional pdfs is cumbersome because

- the conditional pdfs must satisfy some consistency conditions [Gem 84], which cannot be easily verified,
- computation of the joint pdf $p(\mathbf{z})$ from the local conditional pdfs is not straightforward, and
- the relationship between the local spatial characteristics of a realization and the form of the local conditional pdf is not obvious.

Fortunately, every MRF can be described by a Gibbs distribution (hence the name Gibbs random field – GRF), which apparently overcomes these problems.

In order to define a Gibbs distribution, we need to define a clique. A clique $C$, defined over the lattice $\Lambda$ with respect to the neighborhood system $\boldsymbol{N}$, is a subset of $\Lambda$ ($C \subset \Lambda$) such that either $C$ consists of a single site or all pairs of sites in $C$ are neighbors. The set of cliques for the 4-pixel and 8-pixel neighborhoods are shown in Figure B.1. Notice that the number of diferent cliques grows quickly as the number of sites in the neighborhood increases. The set of all cliques is denoted by $\boldsymbol{C}$.

## B.1.2 Gibbs Random Fields

The Gibbs distribution, with a neighborhood system $\boldsymbol{N}$ and the associated set of cliques $\boldsymbol{C}$, is defined for *discrete-valued* random fields as

$$p(\mathbf{z}) = \frac{1}{Q} \sum_{\Omega} e^{-\frac{U(\mathbf{z}=\Omega)}{T}} \delta(\mathbf{z}-\Omega) \tag{B.1}$$

where $\delta(\cdot)$ denotes a Dirac delta function, and the normalizing constant $Q$, called the partition function, is given by

$$Q = \sum_{\Omega} e^{-\frac{U(\mathbf{z}=\Omega)}{T}}$$

and, for continuous-valued random fields as

$$p(\mathbf{z}) = \frac{1}{Q} e^{-\frac{U(\mathbf{z})}{T}} \tag{B.2}$$

where the normalizing constant $Q$ is given by

$$Q = \int_{\Omega} e^{-\frac{U(\mathbf{z})}{T}} \, d\mathbf{z}$$

and $U(\mathbf{z})$, the Gibbs potential (Gibbs energy), is defined as

$$U(\mathbf{z}) = \sum_{C \in \boldsymbol{C}} V_C\left(z(\mathbf{x}), \mathbf{x} \in C\right) \tag{B.3}$$

for both the discrete and continuous-valued random fields. Each $V_C(z(\mathbf{x}), \mathbf{x} \in C)$, called the clique potential, depends only on the values $z(\mathbf{x})$ for which $\mathbf{x} \in C$. The parameter $T$, known as the temperature, is used to control the peaking of the distribution. Note that Gibbs distribution is an exponential distribution that includes Gaussian as a special case.

Gibbs distribution is a joint pdf of all random variables composing the random field, as opposed to a local conditional pdf. It can be specified in terms of certain desired structural properties of the field that are modeled through the clique potentials, which is demonstrated in Section B.2.

### B.1.3 Equivalence of MRF and GRF

The equivalence of an MRF and a GRF is stated by the Hammersley–Clifford theorem, which provides a simple and practical way to specify MRFs through Gibbs potentials.

#### Hammersley–Clifford (H–C) Theorem

Let $\boldsymbol{N}$ be a neighborhood system. Then $z(\mathbf{x})$ is an MRF with respect to $\boldsymbol{N}$ if and only if $p(\mathbf{z})$ is a Gibbs distribution with respect to $\boldsymbol{N}$.

The H–C theorem was highlighted by Besag [Bes 74], based on Hammersley and Clifford's unpublished paper. Spitzer [Spi 71] also provided an alternate proof of the H–C theorem. The work of Geman and Geman [Gem 84] pioneered the popular use of Gibbs distributions to specify MRF models. Section B.2 demonstrates how to select clique potentials for continuous and discrete valued GRFs, and how to specify a Gibbs distribution by using clique potentials.

## B.2 Gibbs Distribution as an *a priori* pdf Model

The Gibbs distribution has been used as a popular *a priori* pdf model to impose the spatial-smoothness constraint in motion estimation (Chapter 4) and image/video segmentation (Chapter 5). This section demonstrates how a spatial-smoothness constraint can be formulated as an *a priori* pdf in the form of a Gibbs distribution. The clique potentials effectively express the local interaction between pixels and can be assigned arbitrarily, unlike the local pdfs in MRFs, which must satisfy certain consistency conditions.

### Example: Case of a Continuous-Valued GRF

A real-valued motion-vector field can be modeled by a continuous-valued GRF. Let us employ a four-point neighborhood system, depicted in Figure B.1, with two-pixel cliques. For continuous-valued GRF, a suitable potential function for the two-pixel cliques may be

$$V_C\left(\mathbf{d}(\mathbf{x}_i),\mathbf{d}(\mathbf{x}_j)\right)=||\mathbf{d}(\mathbf{x}_i)-\mathbf{d}(\mathbf{x}_j)\,||^2 \tag{B.4}$$

where $\mathbf{x}_i$ and $\mathbf{x}_j$ denote the elements of any two-pixel clique, and $||\cdot||$ is the Euclidian distance. In Eqn. (B.3), $V_C(\mathbf{d}(\mathbf{x}_i),\mathbf{d}(\mathbf{x}_j))$ needs to be summed over all two-pixel cliques. Clearly, a spatial configuration of motion vectors with larger potential would have a smaller *a priori* probability.

### Example: Case of a Discrete-Valued GRF

If the motion vectors are quantized, say to 0.5-pixel accuracy, or we are modeling a segmentation label field, then we have a discrete-valued GRF. Suppose that a discrete-valued GRF $\mathbf{z}$ is defined over the $4 \times 4$ lattice, shown in Figure B.2(a), and Figure B.2(b) and (c) show two realizations of a $4 \times 4$ binary segmentation label field $\mathbf{z}$.

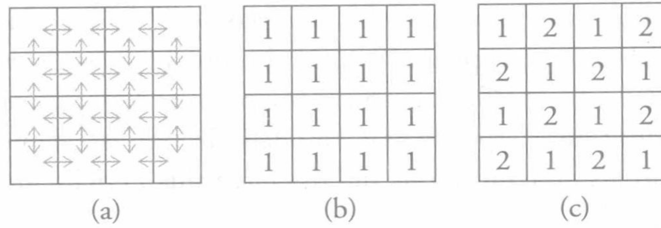



**Figure B.2** Demonstration of a discrete-valued Gibbs model: (a) a generic 4×4 lattice, (b) a realization of the label field $\boldsymbol{z}$, (c) another realization of the label field $\boldsymbol{z}$.

Let the two-pixel clique potential be defined as

$$V_C\left(z(\mathbf{x}_i), z(\mathbf{x}_j)\right) = \begin{cases} -\beta & \text{if } z(\mathbf{x}_i) = z(\mathbf{x}_j) \\ \beta & \text{otherwise} \end{cases} \tag{B.5}$$

where $\beta$ is a positive number.

There are a total of 24 two-pixel cliques in a $4 \times 4$ image (shown by double arrows). It can be easily seen, by summing all clique potentials, that the confgurations shown in Figures B.2(b) and (c) have the Gibbs poten - tials $-24\ \beta$ and $+24\ \beta$, respectively. Clearly, when these Gibbs potentials are substituted in (B.3) and then (B.1), the spatially smooth confguration depicted in Figure B.2(b) has a higher *a priori* probability.

Once a Gibbs distribution (which is a joint pdf) is specified, the local conditional pdf induced by this Gibbs distribution can be easily computed as shown below.

## B.3 Computation of Local Conditional Probabilities from a Gibbs Distribution

In certain applications, such as in the Gibbs sampler method of optimization (see Appendix C), it may be desirable to obtain the local conditional pdfs from the joint pdf given by a Gibbs distribution. Derivation of the local conditional pdfs of a discrete-valued GRF is shown in the following. Starting with the Bayes rule,

$$p(z(\mathbf{x}_i) \mid z(\mathbf{x}_j),\ \text{all}\ \mathbf{x}_j \neq \mathbf{x}_i) = \frac{p(\mathbf{z})}{p(z(\mathbf{x}_j),\ \text{all}\ \mathbf{x}_j \neq \mathbf{x}_i)}$$

$$= \frac{p(z(\mathbf{x}))}{\sum_{z(\mathbf{x}_i)\in\Gamma} p(\mathbf{z})},\ \mathbf{x} \in \Lambda \tag{B.6}$$

where the second line follows from the total probability rule. In particular, let $A_\gamma$ denote the event $z(\mathbf{x}_i) = \gamma$, for all $\gamma \in \Gamma$, and $B$ stand for a fixed realization of the remaining sites $\mathbf{x}_j \neq \mathbf{x}_i$; then (B.6) is simply a restatement of

$$P(A_\gamma \mid B) = \frac{P(A_\gamma \cap B)}{\sum_{\gamma\in\Gamma} P(B \mid A_\gamma)P(A_\gamma)}$$

where $P(\cdot)$ denotes probability (obtained by integrating the pdf $p(\cdot)$ over a range).

Substituting the Gibbs distribution for $p(\cdot)$ in (B.6), the local conditional pdf can be expressed in terms of the clique potentials (after some algebra) as

$$p(z(\mathbf{x}_i) \mid z(\mathbf{x}_j),\ \text{all}\ \mathbf{x}_j \neq \mathbf{x}_i) = Q_{\mathbf{x}_i}^{-1}\ e^{-\frac{1}{T}\sum_{C|\mathbf{x}_i\in C} V_C(z(\mathbf{x})|\,\mathbf{x}\in C)} \tag{B.7}$$

where

$$Q_{\mathbf{x}_i} = \sum_{z(\mathbf{x}_i)\in\Gamma} e^{-\frac{1}{T}\sum_{C|\mathbf{x}_i\in C} V_C(z(\mathbf{x})|\mathbf{x}\in C)}$$

For a more detailed treatment of MRFs and GRFs the reader is referred to [Gem 84]. Vigorous treatment of the statistical formulations can also be found in [Bes 74] and [Spi 71].

## References

[Bes 74] Besag, J., “Spatial interaction and the statistical analysis of lattice systems,” J. Royal Stat. Soc. B, vol. 36, no. 2, pp. 192–236, 1974.

[Che 93] Chellappa, R., and A. Jain, eds., *Markov Random Fields: Theory and Application,* Boston, MA: Academic Press, 1993.

[Cro 83] Cross, G. R., and A. K. Jain, “Markov random field texture models,” IEEE Trans. Patt. Anal. Mach. Intel., vol. PAMI-5, pp. 25–39, Jan. 1983.

[Der 87] Derin, H., and H. Elliott, "Modeling and segmentation of noisy and textured images using Gibbs random field," IEEE Trans. on Patt. Anal. Mach. Intel., vol. 9, pp. 39–55, Jan. 1987.

[Dub 93] Dubois, E., and J. Konrad, "Estimation of 2-D motion fields from image sequences with application to motion-compensated processing," in Motion Analysis and Image Sequence Processing, M. I. Sezan and R. L. Lagendijk, eds., Kluwer, 1993.

[Gem 84] Geman, S., and D. Geman, "Stochastic relaxation, Gibbs distribution, and Bayesian restoration of images," IEEE Trans. on Patt. Anal. Mach. Intel., vol. 6, pp. 721–741, Nov. 1984.

[Pap 92] Pappas, T. N., "An adaptive clustering algorithm for image segmentation," IEEE Trans. Signal Proc., vol. SP-40, pp. 901–914, April 1992.

[Spi 71] Spitzer, F., "Markov random fields and Gibbs ensembles," Amer. Math. Mon., vol. 78, pp. 142–154, Feb. 1971.

APPENDIX C

# Optimization Methods

Several motion-estimation and segmentation problem formulations require minimization of a non-convex criterion function $E(\mathbf{u})$, where $\mathbf{u}$ is some $N$-dimensional unknown vector. Then, the motion-estimation/segmentation problem can be posed so as to find

$$\hat{\mathbf{u}} = \arg\min_{\mathbf{u}} E(\mathbf{u})$$

This minimization is exceedingly difficult due to large dimensionality of the unknown vector and the presence of local minima. With non-convex functions, gradient descent methods (reviewed in Section C.1) generally cannot reach the global minimum, because they get trapped in the nearest local minimum. In Section C.2, we present two simulated (stochastic) annealing algorithms, the Metropolis algorithm [Met 53] and the Gibbs sampler [Gem 84], which are capable of finding the global minimum at the expense of significant increase in computation time. Next, we present three greedy methods that are deterministic approximations to simulated annealing: iterative conditional modes (ICM) [Bes 74], mean-field annealing [Bil 91a], and the highest-confidence-first (HCF) [Cho 90] algorithms to obtain faster convergence in Sections C.3. For a detailed survey of popular annealing procedures the reader is referred to [Kir 83, Laa 87].

## C.1 Gradient-Based Optimization

A function $f(u_1, \ldots, u_n)$ of several unknowns can be minimized by calculating its partials with respect to each unknown, setting them equal to zero, and solving the resulting equations

$$\begin{aligned} \frac{\partial f(\mathrm{u})}{\partial u_1} &= 0 \\ &\vdots \\ \frac{\partial f(\mathbf{u})}{\partial u_n} &= 0 \end{aligned} \tag{C.1}$$

simultaneously for $u_1, \ldots, u_n$. This set of simultaneous equations can be expressed as a vector equation,

$$\nabla_{\mathbf{u}} f(\mathbf{u}) = \mathbf{0} \tag{C.2}$$

where $\nabla_{\mathbf{u}}$ is the gradient operator with respect to the unknown vector $\mathbf{u}$. Because it is difficult to define a closed-form criterion function $f(\mathbf{u})$ for motion estimation or image segmentation, to solve the set of equations (C.2) in closed form, we resort to iterative (numerical) methods.

### C.1.1 Steepest-Descent Method

Steepest descent is probably the simplest numerical optimization method. It updates the present estimate of the location of the minimum in the direction of the negative gradient, called the steepest-descent direction. Since the gradient vector points in the direction of the maximum, the direction of steepest descent is just the opposite direction, which is illustrated in Figure C.1.

In order to get closer to the minimum, we update our current estimate as

$$\mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} - \alpha \, \nabla_{\mathbf{u}} f(\mathbf{u}) \big|_{\mathbf{u}^{(k)}} \tag{C.3}$$

where $\alpha$ is some positive scalar, known as the step-size. The step-size is critical for the convergence of the iterations, because if $\alpha$ is too small, we move by a very small amount each time, and the iterations will take too long to converge. On the other hand, if it is too large the algorithm may become unstable and oscillate about

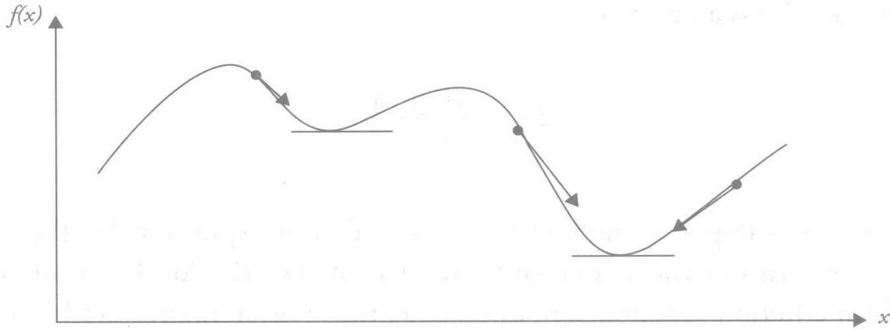



**Figure C.1** Illustration of local minima and the gradient-descent method.

the minimum. In the method of steepest descent, the step-size is usually chosen heuristically.

## C.1.2 Newton–Raphson Method

The optimum value for the step-size $\alpha$ can be estimated using the well-known Newton–Raphson method for root finding. Here, the derivation for a function of a single variable is shown for simplicity. In one unknown, we would like to find a root of $f'(u)$. To this effect, we expand $f'(u)$ in a Taylor series about the point $u^{(k)}$ as

$$f'\left(u^{(k+1)}\right) = f'\left(u^{(k)}\right) + \left(u^{(k+1)} - u^{(k)}\right)f''\left(u^{(k)}\right)$$

Since we wish $u^{(k+1)}$ to be a zero of $f'(u)$, we set

$$f'\left(u^{(k)}\right) + \left(u^{(k+1)} - u^{(k)}\right)f''\left(u^{(k)}\right) = 0 \qquad \text{(C.4)}$$

Solving (C.4) for $u^{(k+1)}$, we have

$$u^{(k+1)} = u^{(k)} - \frac{f'\left(u^{(k)}\right)}{f''\left(u^{(k)}\right)}$$

This result can be generalized for the case of a function of several unknowns as

$$\mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} - \mathrm{H}^{-1}\, \nabla_{\mathbf{u}} f(\mathbf{u})\big|_{\mathbf{u}^{(k)}} \qquad \text{(C.5)}$$

where **H** is the Hessian matrix

$$\mathbf{H}_{ij} = \frac{\partial^2 f(\mathbf{u})}{\partial u_i \, \partial u_j}$$

The Newton–Raphson method finds an analytical expression for the step-size parameter in terms of the second-order partials of the criterion function. When a closed-form criterion function is not available, the Hessian matrix can be estimated by using numerical methods [Fle 87].

The gradient-descent methods suffer from a serious drawback: the solution depends on the initial point. If we start in a "valley," it will be stuck at the bottom of that valley, which may be a "local" minimum, as depicted in Figure C.1. Because the gradient vector is zero or near zero, at or around a local minimum, the updates become too small to move out of a local minimum. One solution to this problem is to initialize the algorithm at several different starting points, and then choose the solution that gives the smallest criterion function. More sophisticated methods to reach the global minimum regardless of the starting point, such as simulated annealing (SA), are discussed next. However, SA methods require significantly more processing time.

## C.2 Simulated Annealing

Simulated annealing (SA) refers to a class of stochastic relaxation algorithms known as Monte Carlo methods. They are essentially prescriptions for a partially random search of the solution space. At each step of the algorithm, the previous solution is subjected to a random perturbation. Unlike deterministic gradient-based iterative algorithms, which always move in the direction of decreasing criterion function, simulated annealing permits, on a random basis, changes that increase the criterion function. This is because an uphill move is sometimes necessary in order to prevent the solution from settling in a local minimum. The probability of accepting uphill moves is controlled by a temperature parameter. The simulated annealing process starts by first "melting" the system at a high enough temperature that almost all random moves are accepted. Then the temperature is lowered slowly according to a "cooling" regime. At each temperature, the simulation must proceed long enough for the system to reach a "steady state." The sequence of temperatures and the number of perturbations at each temperature constitute the "annealing schedule." The convergence of the procedure is strongly related to the annealing schedule. In their pioneering work, Geman and Geman [Gem 84] proposed the temperature schedule

$$T = \frac{\tau}{\ln(i+1)}, \quad i = 1, \ldots \tag{C.6}$$

where $\tau$ is a constant and $i$ is the iteration cycle. This schedule is overly conservative but guarantees reaching the global minimum. Schedules that lower the temperature at a faster rate have also been shown to work (without a proof of convergence).

The process of generating random perturbations is referred to as sampling the solution space. In the following, we present two algorithms that differ in the way they sample the solution space.

### C.2.1 Metropolis Algorithm

In Metropolis sampling, at each step of the algorithm a new candidate solution is generated at random. If this new solution decreases the criterion function, it is always accepted; otherwise, it is accepted according to an exponential probability distribution. The probability $P$ of accepting the new solution is then given by

$$P = \begin{cases} e^{-\frac{\Delta E}{T}} & \text{if } \Delta E > 0 \\ 1 & \text{if } \Delta E \leq 0 \end{cases}$$

where $\Delta E$ is the change in the criterion function due to the perturbation, and $T$ is the temperature parameter. If $T$ is relatively large, the probability of accepting a positive energy change is higher than when $T$ is small for a given $\Delta E$. We provide a summary of the Metropolis algorithm in the following [Met 53]:

1. Set $i = 0$ and $T = T_{max}$. Choose an initial $\mathbf{u}^{(0)}$ at random.
2. Generate a new candidate solution $\mathbf{u}^{(i+1)}$ at random.
3. Compute $\Delta E = E(\mathbf{u}^{(i+1)}) - E(\mathbf{u}^{(i)})$.
4. Compute $P$ from

$$P = \begin{cases} e^{-\frac{\Delta E}{T}} & \text{if } \Delta E > 0 \\ 1 & \text{if } \Delta E \leq 0 \end{cases}$$

5. If $P = 1$, accept the perturbation; otherwise, draw a random number that is uniformly distributed between 0 and 1. If the number drawn is less than $P$, accept the perturbation.

6. Set $i = i+1$. If $i \leq I_{max}$, where $I_{max}$ is predetermined, go to 2.
7. Set $i = 0$, and $\mathbf{u}^{(0)} = \mathbf{u}^{(I_{max})}$. Reduce $T$ according to a temperature schedule. If $T > T_{min}$, go to 2; otherwise, terminate.

Because the candidate solutions are generated by random perturbations, the algorithm typically requires a large number of iterations for convergence. Thus, the computational load of simulated annealing is significant, especially when the set of allowable values $\Gamma$ (defned in Appendix B for $\mathbf{u}$ discrete) contains a large number of values or $\mathbf{u}$ is a continuous variable. Also, the computational load increases with the number of components in the unknown vector.

### C.2.2 Gibbs Sampler

Let's assume that $\mathbf{u}$ is a random vector composed of lexicographic ordering of the elements of a scalar GRF $u(\mathbf{x})$. In Gibbs sampling, the perturbations are generated according to local conditional probability density functions (pdfs) derived from the given Gibbsian distribution, according to (B.7) in Appendix B, rather than making totally random perturbations and then deciding whether or not to accept them. The Gibbs sampler method can be summarized as:

1. Set $T = T_{max}$. Choose an initial $\mathbf{u}$ at random.
2. Visit each site $\mathbf{x}$ to perturb the value of $\mathbf{u}$ at that site as follows:
   a. At site $\mathbf{x}$, compute the conditional probability of $u(\mathbf{x})$ to take each of the allowed values from the set $\Gamma$, given the clique potentials and present values of its neighbors using (B.7).
   b. Once the probabilities for all elements of the set $\Gamma$ are computed, draw the new value of $u(\mathbf{x})$ from this distribution. We clarify the meaning of "draw" by using an example.

**Example.** Suppose that $\Gamma = \{0,1,2,3\}$, and it was found that

$$P(u(\mathbf{x}_i) = 0 \mid u(\mathbf{x}_j), \mathbf{x}_j \in N_{\mathbf{x}_i}) = 0.2$$

$$P(u(\mathbf{x}_i) = 1 \mid u(\mathbf{x}_j), \mathbf{x}_j \in N_{\mathbf{x}_i}) = 0.1$$

$$P(u(\mathbf{x}_i) = 2 \mid u(\mathbf{x}_j), \mathbf{x}_j \in N_{\mathbf{x}_i}) = 0.4$$

$$P(u(\mathbf{x}_i) = 3 \mid u(\mathbf{x}_j), \mathbf{x}_j \in N_{\mathbf{x}_i}) = 0.3$$

Then a random number $R$, uniformly distributed between 0 and 1, is generated, and the value of $u(\mathbf{x}_i)$ is decided as follows:

$$\text{if } 0 \leq R \leq 0.2 \text{ then } u(\mathbf{x}_i) = 0$$

$$\text{if } 0.2 \leq R \leq 0.3 \text{ then } u(\mathbf{x}_i) = 1$$

$$\text{if } 0.3 \leq R \leq 0.7 \text{ then } u(\mathbf{x}_i) = 2$$

$$\text{if } 0.7 \leq R \leq 1 \text{ then } u(\mathbf{x}_i) = 3$$

3. Repeat step 2 sufficiently many times at a given temperature, then lower the temperature, and go to 2. Note that the conditional probabilities depend on the temperature parameter.

Perturbations through Gibbs sampling lead to very interesting properties, which have been shown by Geman and Geman [Gem 84]:

- For any initial estimate, Gibbs sampling will yield a distribution that is asymptotically Gibbsian, with the same properties as the Gibbs distribution used to generate it. This result can be used to simulate a Gibbs random field.
- For the particular temperature schedule (C.6), the global optimum will be reached. However, in practice, convergence with this schedule may be too slow.

## C.3 Greedy Methods

Greedy optimization methods look for simple solutions to complex problems by a step-by-step procedure where the solution that provides the most benefit is chosen at each step. They can be considered as deterministic approximations to simulated annealing. We discuss three such methods: iterated conditional modes, mean-field annealing, and highest confidence first.

### C.3.1 Iterated Conditional Modes

Iterated conditional modes (ICM) algorithm, also known as the greedy algorithm, is a deterministic procedure that aims to reduce the computational load of the stochastic annealing methods. It can be posed as special cases of both the Metropolis and Gibbs sampler algorithms. ICM can best be conceptualized as the "instant freezing" case of the Metropolis algorithm, i.e., when the temperature $T$ is set equal to zero for all iterations. Then the probability of accepting perturbations that increase the value

of the cost function is always 0 (refer to step 4 of the Metropolis algorithm). Alternatively, it has been shown that ICM converges to the solution that maximizes the local conditional probabilities given by (B.7) at each site. Hence, it can be implemented as in Gibbs sampling, but by choosing the value at each site that gives the maximum local conditional probability rather than drawing a value based on the conditional probability distribution.

ICM converges much faster than the SA algorithms. However, because ICM only allows those perturbations yielding negative $\Delta E$, it is likely to get trapped in a local minimum, much like gradient-descent algorithms. Thus, it is critical to initialize ICM with a reasonably good initial estimate. The use of ICM has been reported for image restoration [Bes 74] and image segmentation [Pap 92].

## C.3.2 Mean-Field Annealing

Mean-field annealing is based on the "mean-field approximation" (MFA) idea in statistical mechanics. MFA allows replacing each random variable (random field evaluated at a particular site) by the mean of its marginal probability distribution at a given temperature. Then mean-field annealing is concerned about the estimation of these means at each site. Because the estimation of each mean is dependent on using the neighboring sites, this estimation is performed using an annealing schedule. The algorithm for annealing the mean field is similar to SA except that stochastic relaxation at each temperature is replaced by a deterministic relaxation to minimize the so-called mean field error, usually using a gradient-descent algorithm.

Historically, MFA was limited to Ising-type models described by a criterion function involving a binary vector. It was later extended to a wider class of problems, including those with continuous variables [Bil 91b]. Experiments suggest that the MFA is valid for MRFs with local interactions over small regions. Thus, computations of the means and the mean-field error are often based on Gibbsian distributions. It has been claimed that mean-field annealing converges to an acceptable solution approximately 50 times faster than SA. The implementation of MFA is not unique. Covering different implementations of mean-field annealing [Orl 85, Bil 92, Abd 92, Zha 93] is beyond the scope of this book.

## C.3.3 Highest Confidence First

The highest-confidence-first (HCF) algorithm proposed by Chou and Brown [Cho 90] is a deterministic, non-iterative algorithm. It is guaranteed to reach a local minimum of the potential function after a finite number of steps.

In the case of a discrete-valued GRF, the minimization is performed on a site-by-site basis according to the following rules: i) Sites with reliable data can be labeled without using the *a priori* probability model. ii) Sites where the data is unreliable should rely on neighborhood interaction for label assignment. iii) Sites with unreliable data should not affect sites with reliable data through neighborhood interaction.

Guided by these principles, a scheme that determines a particular order for assigning labels and systematically increases neighborhood interaction is designed. Initially, all sites are labeled "uncommitted." Once a label is assigned to an uncommitted site, the site is committed and cannot return to the uncommitted state. However, the label of a committed site can be changed through another assignment. A "stability" measure is calculated for each site based on the local conditional *a posteriori* probability of the labels at that site, to determine the order in which the sites are to be visited. The procedure terminates when the criterion function can no longer be decreased by reassignment of the labels.

Among the deterministic methods, HCF is simpler and more robust than MFA, and more accurate than the ICM. Extensions of HCF for the case of continuous variables also exist.

## References

[Abd 92] Abdelqader, I., S. Rajala, W. Snyder, and G. Bilbro, "Energy minimization approach to motion estimation," Signal Processing, vol. 28, pp. 291–309, Sep. 1992.

[Bes 74] Besag, J., "Spatial interaction and the statistical analysis of lattice systems," J. Royal Stat. Soc. B, vol. 36, no. 2, pp. 192–236, 1974.

[Bil 91a] Bilbro, G. L., and W. E. Snyder, "Optimization of functions with many minima," IEEE Trans. Syst., Man and Cyber., vol. 21, pp. 840–849, Jul/Aug. 1991.

[Bil 91b] Bilbro, G. L., W. E. Snyder, and R. C. Mann, "Mean field approximation minimizes relative entropy," J. Opt. Soc. Am. A, vol. 8, pp. 290–294, Feb. 1991.

[Bil 92] Bilbro, G. L., W. E. Snyder, S. J. Garnier, and J. W. Gault, "Mean field annealing: A formalism for constructing GNC-like algorithms," IEEE Trans. Neural Networks, vol. 3, pp. 131–138, Jan. 1992.

[Cho 90] Chou, P. B., and C. M. Brown, "The theory and practice of Bayesian image labeling," Int. J. Comp. Vis., vol. 4, pp. 185–210, 1990.

[Fle 87] Fletcher, R., *Practical Methods of Optimization, Second Edition*, New York, NY: Wiley, 1987.

[Gem 84] Geman, S., and D. Geman, "Stochastic relaxation, Gibbs distribution, and Bayesian restoration of images," IEEE Trans. Patt. Anal. Mach. Intel., vol. 6, pp. 721–741, Nov. 1984.

[Kir 83] Kirkpatrick, S., C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," Science, vol. 220, pp. 671–679, May 1983.

[Laa 87] Van Laarhoven, P.J.M. and E. H. Aarts, *Simulated Annealing: Theory and Applications,* Dordrecht, Holland: Reidel, 1987.

[Met 53] Metropolis, N., A. Rosenbluth, M. Rosenbluth, H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," J. Chem. Phys., vol. 21, pp. 1087–1092, Ju 1953.

[Orl 85] Orland, H., "Mean-field theory for optimization problems," J. Physique Lett., vol. 46, no. 17, pp. 763–770, 1985.

[Pap 92] Pappas, T. N., "An adaptive clustering algorithm for image segmentation," IEEE Trans. Signal Proc., vol. SP-40, pp. 901–914, April 1992.

[Zha 93] Zhang, J., and J. Hanauer, "The mean field theory for image motion estimation," Proc. IEEE Int. Conf. ASSP, vol. 5, pp. 197–200, Minneapolis, MN, 1993.

APPENDIX D

# Model Fitting

Several motion-estimation, 3D-structure estimation, and image/motion-segmentation problems require fitting a model to available data samples. This appendix presents model-fitting solutions that are common to these problems. Linear least-squares or total least-squares methods are used when available data samples are free from outliers. Random-sample consensus (RANSAC) is an effective method to deal with data with outliers.

## D.1 Least-Squares Fitting

In order to keep the presentation simple, let's suppose that the available data consists of pairs of points $(x_1,y_1),\ldots,(x_N,y_N)$ and we want to fit a line $y = m\,x + b$ to these points such that sum of squared vertical distances between the given points and the line $\delta_i = y - y_i$, depicted in Figure D.1(a), is minimized. Hence, the problem can be stated as: given $(x_1,y_1),\ldots,(x_N,y_N)$, find $(m, b)$ to minimize

$$E_{LS} = \sum_{i=1}^{N} (y_i - m\,x_i - b)^2 \tag{D.1}$$

which can be expressed in vector-matrix form as the least-squares solution of a set of linear equations of the form $\mathbf{A}\,\mathbf{h} = \mathbf{y}$ to minimize

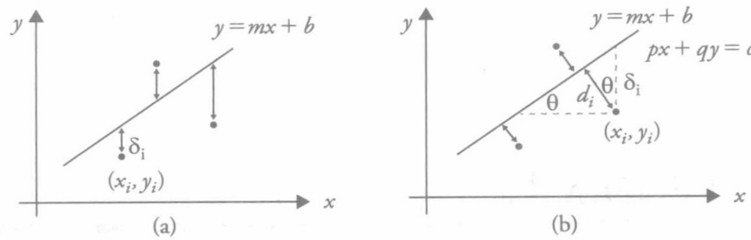



**Figure D.1** Straight-line fitting: (a) least-squares fitting and (b) total least-squares fitting.

$$E_{LS} = \sum_{i=1}^{N}\left(y_i - [x_i \;\; 1]\begin{bmatrix} m \\ b \end{bmatrix}\right)^2 = \left\| \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} - \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_N & 1 \end{bmatrix}\begin{bmatrix} m \\ b \end{bmatrix} \right\|^2 = \| \mathbf{y} - \mathbf{A}\ \mathbf{h} \|^2$$

where $\mathbf{y}$ denotes the vector of observations, $\mathbf{h}$ is the vector of unknowns, and $\mathbf{A}$ is the matrix of coefficients as defined in the equation. Since

$$E_{LS} = \|\mathbf{y} - \mathbf{A}\ \mathbf{h}\|^2 = (\mathbf{y} - \mathbf{A}\ \mathbf{h})^{\mathrm{T}}(\mathbf{y} - \mathbf{A}\ \mathbf{h}) = \mathbf{y}^{\mathrm{T}}\mathbf{y} - 2(\mathbf{A}\ \mathbf{h})^{\mathrm{T}}\mathbf{y} + (\mathbf{A}\ \mathbf{h})^{\mathrm{T}}(\mathbf{A}\ \mathbf{h})$$

we have

$$\frac{dE_{LS}}{d\mathbf{h}} = 2\ \mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{h} - 2\mathbf{A}^{\mathrm{T}}\mathbf{y} = \mathbf{0} \tag{D.2}$$

which gives

$$\hat{\mathbf{h}}_{LS} = (\mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{y} \tag{D.3}$$

## D.2 Least-Squares Solution of Homogeneous Linear Equations

Given a set of $M$ homogeneous linear equations in $N$ unknowns, $M > N$, in the form

$$\mathbf{A}\ \mathbf{h} = \mathbf{0} \tag{D.4}$$

where $\mathbf{A}$ is an $M \times N$ matrix of coefficients with rank $N$ and $\mathbf{h}$ is an $N \times 1$ vector consisting of unknown model parameters. Assuming the coefficient matrix is noisy, the over-determined system $\mathbf{A}\,\mathbf{h} = \mathbf{0}$ is inconsistent and does not have a solution.

We can define a least-squares solution, given by

$$\hat{\mathbf{h}} = \arg\min_{\mathbf{h}} \|\mathbf{A}\,\mathbf{h}\|, \text{ subject to } \|\mathbf{h}\| = 1$$

The constraint $\|\mathbf{h}\| = \mathbf{1}$ avoids the trivial solution $\mathbf{h} = \mathbf{0}$ and sets an arbitrary scale factor. We can reexpress the constraint as

$$1 - \mathbf{h}^T\mathbf{h} = 0$$

and using the Lagrangian formulation, the constraint-optimization problem can be expressed as

$$\hat{\mathbf{h}} = \arg\min_{\mathbf{h}} \{\mathbf{h}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\,\mathbf{A}\,\mathbf{h} + \lambda(1 - \mathbf{h}^T\mathbf{h})\}$$

where the solution can be found by solving the equation

$$\frac{d}{d\,\mathbf{h}}\left\{\mathbf{h}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\;\mathbf{A}\;\mathbf{h} + \lambda(1 - \mathbf{h}^T\mathbf{h})\right\} = 0$$

which yields

$$\mathbf{A}^{\mathrm{T}}\,\mathbf{A}\,\mathbf{h} + \lambda\mathbf{h} = 0$$

Hence, we know that $\mathbf{h}$ is an eigenvector of $\mathbf{A}^{\mathrm{T}}\,\mathbf{A}$ and $\lambda$ is an eigenvalue, and the least-squares error is given by

$$e = \mathbf{h}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\,\mathbf{A}\,\mathbf{h} = \mathbf{h}^{\mathrm{T}}\lambda\mathbf{h}$$

Therefore, the error will be minimal for $\lambda = \min_i \lambda_i$ and the solution $\hat{\mathbf{h}}$ is the eigenvector of $\mathbf{A}^{\mathrm{T}}\,\mathbf{A}$ corresponding to the smallest eigenvalue.

### D.2.1 Alternate Derivation

An alternate derivation can be based on the singular value decomposition (SVD) of the matrix $\mathbf{A}$, $\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\mathrm{T}}$, where $\mathbf{U}$ is an $M \times N$ orthonormal matrix, $\boldsymbol{\Sigma}$ is an $N \times N$

diagonal matrix, and $\mathbf{U}$ is an $N \times N$ orthonormal matrix. Since $\mathbf{U}$ and $\mathbf{V}$ are orthonormal, we have

$$||\mathbf{Ah}|| = ||\mathbf{U\Sigma V^T h}|| = ||\mathbf{\Sigma V^T h}|| \text{ and } ||\mathbf{V^T h}|| = ||\mathbf{h}||$$

If we let $\mathbf{y} = \mathbf{V^T h}$, then $||\mathbf{Ah}|| = ||\mathbf{\Sigma y}||$ we can pose the problem as

$$\hat{\mathbf{y}} = \arg\min_{\mathbf{y}} ||\mathbf{\Sigma y}||, \text{ subject to } ||\mathbf{y}|| = \mathbf{1}$$

Since $\Sigma$ is diagonal and the singular values are sorted in descending order, $\hat{\mathbf{y}} = [0\ 0 \ldots 0\ 1]^T$ and hence the solution $\hat{\mathbf{h}} = (\mathbf{V}^T)^{-1}\hat{\mathbf{y}}$ is given by the last column of the matrix $\mathbf{V}$.

## D.3 Total Least-Squares Fitting

The method of total least-squares fits a line to given data samples by modeling the uncertainty in both variables as $x = x_i + \epsilon$ and $y = y_i + \delta$. This can be achieved by minimizing the sum of squares of the shortest distances $d_i$, depicted in Figure D.1(b), between the points and the line. The shortest distance between a point $(x_i, y_i)$, and the line $y = mx + b$ is given by

$$d_i = \delta_i \cos\theta = \frac{\delta_i}{\sqrt{1 + \tan^2\theta}} = \frac{\delta_i}{\sqrt{1 + m^2}}$$

where the slope of the line $m = \tan\theta$. Hence, the total least-squares problem can be formulated as: find $(m, b)$ to minimize

$$E_{TLS} = \frac{1}{1 + m^2} \sum_{i=1}^{N} (y_i - m\, x_i - b)^2 \tag{D.4}$$

A more convenient formulation of the TLS problem is possible if we parameterize the equation of the line as $px + qy = c$, where $\mathbf{n} = [p\ q]^T$ denotes the unit normal vector such that $p^2 + q^2 = 1$. Then, the minimum distance is given by $|px_i + qy_i - c|$, and the problem can be formulated as find $(p, q, c)$ to minimize the sum of squared distances

$$E_{TLS} = \sum_{i=1}^{N} (px_i + qy_i - c)^2 \tag{D.5}$$

We can first evaluate $c$ in terms of $p$ and $q$ by differentiating the cost function with respect to $c$

$$\frac{dE_{TLS}}{d\,c} = -\sum_{i=1}^{N}(px_i + qy_i - c) = 0 \tag{D.6}$$

to obtain

$$c = p\bar{x} + q\bar{y} \tag{D.7a}$$

where

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \text{ and } \bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i \tag{D.7b}$$

Next, we substitute the value of $c$ (D.7) in the cost function (D.5) to solve for $p$ and $q$

$$E_{TLS} = \sum_{i=1}^{N}\left(p(x_i - \bar{x}) + q(y_i - \bar{y})\right)^2 = \left\| \begin{bmatrix} x_i - \bar{x} & y_i - \bar{y} \\ \vdots & \vdots \\ x_i - \bar{x} & y_i - \bar{y} \end{bmatrix} \begin{bmatrix} p \\ q \end{bmatrix} \right\|^2 = (\mathbf{Ah})^{\mathrm{T}}\,\mathbf{Ah} \tag{D.8a}$$

where

$$\mathbf{A} = \begin{bmatrix} x_i - \bar{x} & y_i - \bar{y} \\ \vdots & \vdots \\ x_i - \bar{x} & y_i - \bar{y} \end{bmatrix} \text{ and } \mathbf{h} = \begin{bmatrix} p \\ q \end{bmatrix} \tag{D.8b}$$

Differentiating $E_{TLS}$ with respect to **h** and setting the result equal to zero,

$$\frac{dE_{TLS}}{d\,\mathbf{h}} = 2\mathbf{A}^{\mathrm{T}}\mathbf{Ah} = 0 \tag{D.9}$$

The solution of $\mathbf{A}^{\mathrm{T}}\mathbf{Ah} = \mathbf{0}$, subject to $||\mathbf{h}||^2 = 1$, is the eigenvector of $\mathbf{A}^{\mathrm{T}}\mathbf{A}$ associated with the smallest eigenvalue (i.e., the least-squares solution to the homogeneous linear system $\mathbf{Ah} = \mathbf{0}$).

An alternative derivation of the TLS solution based on the SVD is presented in [Mar 07]. Note that the singular values of an $M \times N$ real matrix $\mathbf{A}$ are the square roots of the eigenvalues of the $N \times N$ real, symmetric matrix $\mathbf{A}^T\mathbf{A}$, and given the SVD decomposition $\mathbf{A} = \mathbf{U\Sigma V}^T$, the columns of $\mathbf{V}$ correspond to eigenvectors of $\mathbf{A}^T\mathbf{A}$.

## D.4 Random-Sample Consensus (RANSAC)

Random-sample consensus (RANSAC) is a general framework for model fitting in the presence of outliers. It can be summarized by the following steps:

1. Choose a minimal subset of the original data, called the hypothetical inliers set, at random.
2. Fit the model to the selected set of hypothetical inliers.
3. Test how well the remaining data points fit the model. Those points that fit the estimated model well are recorded as members of the *consensus set.*
4. Refine the model by re-estimating it using all members of the consensus set.
5. The estimated model is deemed good if sufficiently many data points have been classified as part of the consensus set. Hence, we keep the refined model if its consensus set is larger than the previously saved model; otherwise, we go to step 1 and repeat the process.

This procedure is repeated $N$ times where $N$ is a fixed number, each time producing either a new refined model with a corresponding consensus set or a rejected model. RANSAC produces a reasonable result with a certain probability, which increases as the number of repetitions $N$ increases [Fis 81]. When $N$ is limited, RANSAC may perform poorly if the number of inliers is less than 50% of the data, and the solution may not be optimal even for moderately contaminated sets. RANSAC can only estimate one model for a given data set. When the data set can be modeled by two (or more) models, the Hough transform is a more robust technique for finding correct models.

## References

[Fis 81] Fischler, M. A., R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," Communications of the ACM, vol 24, pp 381–395, 1981.

[Mar 07] Markovsky, I., and S. Van Huffel, "Overview of total least-squares methods," Signal Processing, vol. 87, no. 10, pp. 2283–2302, October 2007.

# Glossary（术语表）

Above-right predictor (ARP)　右上预测器（用于对当前块的右上角区域运动矢量的获取）

Accommodation distance　调视距离（为使得视网膜上的成像清晰而调整的距离）

Advanced residual prediction (ARP)　高级残差预测

Advanced Television System Committee (ATSC) standard　ATSC 标准（是美国的数字电视国家标准，由美国的高级电视业务顾问委员会（Advanced Television System Committee）于 1995 年 9 月正式通过）

AE (angular error)　角误差

Affine model　仿射模型

Alpha-trimmed mean filters　修正的 α 均值滤波器

AMVP(Advanced-motion vector prediction)　高级运动矢量预测

Anaglyph　补色立体图

Anchor pictures　锚帧图像（AP）（多视点视频编码中的参考帧，编码时不用于空间预测，而只用于视点间预测）

Anti-alias filtering　抗混叠滤波

Aperture problem　孔径问题

Apparent motion　表观运动（在本书中指，通过观察视频感知到的对应场或光流场）

Artifacts　引入失真（原意为人工产品；也可译为伪影，即图像处理过程中引入的各种失真或非理想情况）

ASO (arbitrary slice ordering)　任意条带顺序

Asymmetric property　不对称特性

AWA (adaptive-weighted-averaging) filter　自适应加权均值滤波器

Backward-motion estimation　后向运动估计

Bayer color-filter array pattern　Bayer 彩色滤波阵列模式（指彩色系统中的 RGB 三原色按照 GRGR/BGBG 排列的一种阵列模式，最早由 Bayer 提出，又称为 Bayer color-filter pattern）

Benchmarking　基准

Bi-lateral filters　双边滤波器

Binocular depth cues　双目深度线索

BLA (broken-link access) picture　断链访问图像

Blind-image restoration　图像盲复原

Blur　模糊（由于运动、镜头散焦、噪声等引起的图像锐度降低）

CABAC (context-adaptive binary arithmetic coding)　上下文自适应二进制算术编码

Cartesian coordinates　笛卡尔坐标系

CAVLC (context-adaptive variable-length coding)　上下文自适应变长编码

CCMF (cross-correlated multi-frame) Wiener filter　互相关多帧维纳滤波器

CCN (content centric networking)　内容中心网络

CDF (cumulative distribution function)　累积分布函数

CEA (Consumer Electronics Association)　消费电子协会

Center of projection　投影中心

CFA (color filter array) interpolation　彩色滤波阵列内插

Chrominance　色度

Chunks　（数据）块

CIE (International Commission on Illumination)

国际照明委员会
CLS (constrained least-squares) filtering 有约束最小二乘滤波器
Clustering 聚类
Coiflet filters Coiflet 滤波器（Coiflet 小波系数滤波器）
Contouring artifacts 轮廓伪影
Covariance-based adaptive interpolation 基于协方差的自适应内插
CR (compression ratio) 压缩率
CR (conditional replenishment) 条件补偿
CRA (clean random-access) picture 清除随机访问图像
Critical velocity 临界速度
DBBP (depth-based block partitioning) 基于深度的块分割
DCP (disparity-compensated prediction) 视差补偿预测
Deblurring of images 图像去模糊
Decimation 抽取（信号降采样）
Defocus blur 散焦模糊
Dehomogenization 非均匀化
De-mosaicking 去马赛克
Dense-correspondence estimation problem 稠密对应估计问题
Dense-motion (optical flow/displacement) estimation 稠密运动（光流 / 位移）估计
DFD (displaced-frame difference) 位移帧间差
Diamond search (DS) 钻石搜索
DIBR (depth-image based rendering) 基于深度图像的绘制
Differential methodsl 差分法
Diffusion-based-in-painting 基于扩散的图像修复
Digital dodging-and-burning 数字淡化和加深（不同程度的数字化曝光）
Digital micromirror devices (DMDs) 数字微镜晶片
Digital Terrestrial Multimedia Broadcasting standards 数字多媒体地面广播标准
Direct Linear Transformation (DLT) 直接线性变换
Disocclusion regions 空洞区
Distortion 失真
DMS (discrete memoryless source) 离散无记忆源
DMVP (depth-based motion vector prediction) 基于深度的运动矢量预测
Down-conversion 下转换
Down-sampling (sub-sampling) 亚采样
DVI (Digital Visual Interface) standard 数字视频接口标准
DWT (discrete-wavelet transform) 离散小波变换
EM (expectation-maximization) algorithm 期望最大化算法
Embedded block coding with optimized truncation (EBCOT) 优化截断的嵌入块编码
Encrypted DCP files 加密 DCP 文件
Entropy coding 熵编码
Epipolar geometry 极几何（双目立体匹配中两个透视相机间的一种特殊的几何关系）
ETSI (European Telecommunications Standards Institute) 欧洲电信标准协会
European Broadcasting Union (EBU) 欧洲广播联盟
Exemplar-based methods 基于样例的方法
EZW (embedded zerotree wavelet transform) 嵌入小波零树变换
Fidelity range extensions (FRExt) 保真度范围扩展
Field pictures 场图
Four-fold symmetry 四元对称
FR (full reference metrics) 全参考矩阵
Free-view 2D video 自由视角 2D 视频

Gaze-shifting eye movements 注视转移眼动
Gibbs random field (GRF) Gibbs 随机场
GOP (group of pictures) 图像组
GT (ground-truth) data 标准数据（指得到的原始真实数据）
GVR (gradual view refresh) 渐进刷新
HCF (highest confidence first) algorithm 最大可信度优先算法
HDMI (High-Definition Multimedia Interface) 高清晰度多媒体接口
HDS (HTTP Dynamic Streaming) HTTP 动态流
Hermitian symmetric Hermitian 共轭
HEVC (high-efficiency video-coding) standard 高效视频编码标准（ITU-T 和 MPEG 联合，于 2013 年 1 月 23 日正式发布的视频压缩编码国际标准）
Hexagonal matching 六边形匹配
Histogram 直方图
Homography 单应性（一一对应性）
IC (illumination compensation) 亮度补偿
ICC (International Color Consortium) 国际色彩联盟
ICM (iterated conditional mode) 条件迭代模式
ICT (irreversible color transform) 单向色彩变换
IDD-BM3D (iterative decouple deblurring-BM3D) 迭代弱化去模糊— BM3D
IDR (instantaneous decoding refresh) picture 即时解码刷新图像（H.264 中的 IDR 帧）
Ill-posed problem 病态问题
Illumination 照度
Illumination compensation coding tool 照度补偿编码工具
Image rectification 图像校正
Image sharpening 图像锐化
Image smoothing 图像平滑
Inertial sensing motion tracking 惯性感应运动跟踪
Integer transforms 整数变换
Integer-valued vectors 整数值矢量
Integrated Multimedia Broadcasting (ISDB) standard 综合多媒体广播标准
Interlace lattice 交错栅格
Interlace video input 隔行视频输入
Interlaced scanning 隔行扫描
Interlaced video 隔行制式视频
Interpupilar distance 瞳距
Intersection of two lattices 两个点阵的交集
Intra-frame compression modes 帧内压缩模式
Irregular Repeat-Accumulate codes 不规则重复累积码
Isometry 刚体变换（原意为等距，文中指只有旋转和平移，没有其他变换的仿射模型）
Isomorphic signals 同构信号
Kernel selection 核选择
K-means algorithm K- 均值算法
K-nearest neighbor method K- 近邻方法
Lambertian reflectance model 兰氏反射模型
Lattice(s) 栅格
Least-squares (LS) solution 最小二乘法
Lempel–Ziv coding LZ 编码
Lenticular sheets 透镜片
Lexicographic order 词典顺序
Linear contrast manipulation 线性对比度操作
Linear forward wavelet transform 线性前向小波变换
LMMSE (linear minimum mean-square error) filter 线性最小均方误差滤波器
Log-luminance domain 亮度对数域
LR (low-resolution) frames 低分辨率帧
LSB (least significant bit-plane) 最低有效比特位面
Mach band effect 马赫带效应（对于不同亮度

区域之间的边界区域，人类视觉系统会过高或过低估计其亮度值的现象）

Machine-learning methods 机器学习方法

MAD (minimum mean absolute difference) 最小平均绝对差

Markov random field (MRF) Markov 随机场

Marr-Hildreth scale space theory M-H 尺度空间理论

Masking 蒙版（掩模）

Maximum-likelihood segmentation 最大似然分割

Maxshift method 最大平移法

MC (motion compensation) 运动补偿

MCP (motion-compensated prediction) 运动补偿预测

Mean square difference (MSE) 均方差

Mean-shift (MS) algorithm 均值漂移算法

Mean-square quantization errors 均方量化误差

ML (maximum likelihood) estimate 最大似然估计

MMCO (memory-management control- operation) 内存管理控制操作

Mosaic representation 马赛克表示（指 photo mosaic，即相片镶嵌或马赛克拼图，将多张窄视角照片拼贴成宽视角照片）

Mosquito noise 蚊式噪声

Motion trajectory 运动轨迹

MPC (maximum matching pel count) 最大匹配像素计数

MPEG (Moving Picture Experts Group) 活动图像专家组，也是图像压缩标准的名称

MRF (Markov random field) Markov 随机场

Multiple motions 复杂运动

Multi-resolution frame difference analysis 多分辨率帧间差分分析

Multi-resolution pyramid representations 多分辨率金字塔表示

Multi-scale representation 多尺度表示

Multi-view video coding (MVC) standard 多视点视频编码（MVC）标准

MVD (multi-view-video-plus-depth) format 多视点加深度格式

NLM (non-local means) filtering 非局部均值滤波器

NSHP (Non-symmetric half-plane) support 非对称半平面支撑

Nyquist criterion 奈奎斯特准则

Optical flow 光流

Order-statistics filters 阶统计滤波器

Orthogonal filters 正交滤波器

Perfect-reconstruction (PR) property 完美重构特性

PES (packetized elementary streams) 可打包元素流

PEVQ (perceptual evaluation of video quality) 视频质量感知评估

PM&S (pattern matching and substitution) 模式匹配和替代

Polarizing filters 偏振滤波器

Polyphase implementation of decimation filters 抽取滤波器的多相实现

Projections onto convex sets (POCS) formulation 凸集投影公式

RADL (random access decodable leading) 随机接入可解码引导

RASL (random access skipped leading) 随机接入跳跃引导

Reciprocal lattice 倒易格子

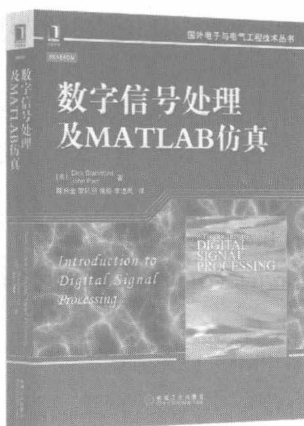
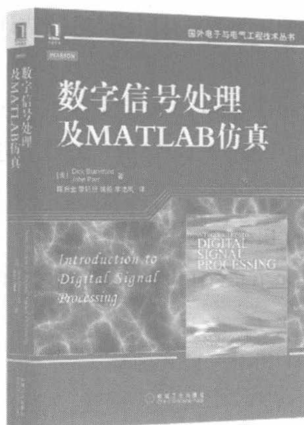
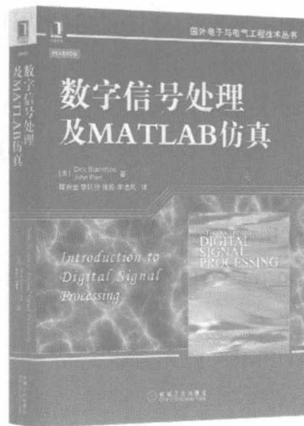
Recognition/example-based methods 基于识别/样例的方法

Recursive filters 递归滤波器

Recursively computable prediction model 可计算递归预测模型

Redundancy reduction　冗余去除
Reversible color transform (RCT)　可逆彩色变换
Ringing artifacts　振铃效应
RLC (run-length coding)　行程编码
ROI (region-of-interest) coding　感兴趣区域编码
RTMP (Real-Time Messaging Protocol)　实时消息协议
RTP (Real-time Transport Protocol)　实时传输协议
RTSP (Real-Time Streaming Protocol)　实时流协议
Run mode　运行模式
SAD (sum of absolute differences)　绝对误差和
SEA (successive elimination algorithm)　逐次消元算法
scale-invariant feature transform (SIFT)　尺度不变特征变换（一般直接写成 SIFT）
SKL (sequential Karhunen–Loeve) algorithm　序列 K-L 算法
Smearing，see Blur　模糊
SMV (super multi-view) displays　超多视角显示
Snakes (active-contour models)　蛇模型（主动轮廓模型）
SNR (signal-to-noise ratio)　信噪比
Sparse representations　稀疏表示
Sparse-correspondence estimation　稀疏相似估计
Statistical redundancy　统计冗余
Stein's unbiased risk estimate (SURE)　Stein 无偏代价估计
Sub-pixel motion estimation　亚像素运动估计
Super-resolution　超分辨率重建
SVC (scalable-video coding)　尺度可变视频编码
SVD (singular-value decomposition)　奇异值分解
Texture coding tools　纹理编码工具
TR (time-recursive) filters　时间递归滤波器
Ultra-high definition television (UHDTV)　超高清电视
Unsharp masking (USM)　虚光蒙版
Up-conversion　上转换
URQ (uniform reconstruction quantization)　归一化重构量化
VGA (Video Graphics Array) display standard　视频图形阵列显示标准
VLC (variable-length source coding)　变长源编码
Wavelet shrinkage　小波收缩
Wavelet transform coding　小波变换编码
Weak-perspective projection model　弱透视投影模型
XGA(Extended Graphics Array)　扩展图形阵列

# 推荐阅读

## 数字信号处理及MATLAB仿真

作者：Dick Blandford 等 译者：陈后金 等 书号：978-7-111-48388-5 定价：95.00元

本书是美国伊凡斯维尔大学电子与计算机工程专业的DSP课程教材，注重理论与应用相结合，前7章重点讲述数字信号处理基础理论和知识，包括DSP的概述、线性信号和系统概念、频率响应、抽样和重建、数字滤波器的分析和设计、多速率DSP系统；后4章侧重于DSP应用，包括数字滤波器的实现、数字音频系统、二维数字信号处理和小波分析。本书可作为电子信息、通信、控制、仪器仪表等相关专业本科生的DSP课程教材，对初级DSP工程师也是一本实用的参考书。

## 数字信号处理及应用

作者：Richard Newbold 等 译者：李玉柏 等 书号：978-7-111-51340-7 定价：119.00元

本书基于真实设备与系统，研究如何进行数字信号处理的软硬件设计与实现，详细阐述了模拟和数字信号调谐、复数到实数的变换、数字信道化器的设计以及数字频率合成技术，并重点讨论了多相滤波器（PPF）、级联的积分梳状（CIC）滤波器、数字信道器等业界常用的一些的信号处理应用。本书适合即将进入信号处理领域的大学毕业生，也适合有一定DSP设计经验的业界工程师阅读。

## 数字信号处理：系统分析与设计（原书第2版）

作者：Paulo S. R. Diniz 等 译者：张太镒 等 ISBN：978-7-111-41475-9 定价：85.00元
英文版 ISBN：978-7-111-38253-9 定价：79.00元

本书全面、系统地阐述了数字信号处理的基本理论和分析方法，详细介绍了离散时间信号及系统、傅里叶变换、z变换、小波分析和数字滤波器设计的确定性数字信号处理，以及多重速率数字信号处理系统、线性预测、时频分析和谱估计等随机数字信号处理，使读者深刻理解数字信号处理的理论和设计方法。本书不仅可以作为高等院校电子、通信、电气工程与自动化、机械电子工程和机电一体化等专业本科生或研究生教材，还可作为工程技术人员DSP设计方面的参考书。